IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

Towards a Robust Sensor Fusion Step for 3D Object Detection on Corrupted Data

Maciej K. Wozniak*[1], Viktor Kårefjärd*[1], Marko Thiel[2], Patric Jensfelt[1]

***Abstract*—Multimodal sensor fusion methods for 3D object detection have been revolutionizing the autonomous driving research field. Nevertheless, most of these methods heavily rely on dense LiDAR data and accurately calibrated sensors which is often not the case in real-world scenarios. Data from LiDAR and cameras often come misaligned due to the miscalibration, decalibration, or different frequencies of the sensors. Additionally, some parts of the LiDAR data may be occluded and parts of the data may be missing due to hardware malfunction or weather conditions. This work presents a novel *fusion step* that addresses data corruptions and makes sensor fusion for 3D object detection more robust. Through extensive experiments, we demonstrate that our method performs on par with state-of-the-art approaches on normal data and outperforms them on misaligned data.**

***Index Terms*—Object Detection, Segmentation and Categorization; Sensor Fusion; Deep Learning for Visual Perception**

I. Introduction

SELF-driving cars must understand their own surroundings, such as vehicles, pedestrians, or cyclists, as well as their pose to further estimate the velocity or future trajectory of moving objects and plan their own movement accordingly. 3D object detection is often used to obtain this semantic information about the environment [1].

3D object detection methods rely on different types of data collected using LiDAR [2] or RGB cameras [3], or a combination of those [4].

Although these methods are capable of achieving impressive results, they often heavily rely on dense LiDAR data and accurately calibrated sensors. Unfortunately, this is often not the case in real-world scenarios and there are different ways the input data can be corrupted. Data from LiDAR and camera often comes misaligned due to poor initial calibration or decalibration throughout the vehicle movement [5] as well as different frequencies or latencies of the sensors [6]. Additionally, some areas of the LiDAR may be occluded and parts of the data may be missing due to hardware malfunction, weather conditions, or reflective surfaces [7]. Furthermore, robotic platforms use LiDAR sensors with different resolutions and even though some methods claim to achieve good results on any given data set, they often underperform when tested on domains they are not trained on (e.g. trained on 64-layer LiDAR, tested on 16 layers) [8].

These different cases of data corruption lead to a considerable decline in performance for state-of-the-art single-modality 3D object detection methods, making them unreliable in real-world scenarios. While multimodal fusion methods are also impacted by these issues, they are more resilient than single-modality approaches. Nevertheless, their effectiveness and robustness heavily depend on where and how the information is fused within the model.

For example, early fusion, which combines modalities almost at the input level, is more prone to corrupted data. On the other hand, deep fusion can be more robust as it allows the network to learn more abstract representations from multiple modalities, potentially mitigating the impact of corrupted or missing information.

Similarly, the way in which the information is fused, what we refer to as *fusion step*, can exhibit different levels of sensitivity to corrupted data. For instance, combining features from different modalities directly (e.g. simply concatenating them together) makes the model highly susceptible to corruption in any of the modalities, whereas using convolution operations to fuse the data can improve handling noise and misalignment.

This work aims to explore how multi-modal fusion can be performed to ensure robustness to corrupted data, required for a practical application, since in the real world we rarely operate on dense data without missing information and with perfectly calibrated sensors. Our main contribution is a *novel fusion step* for 3D object detection that outperforms other proposed state-of-the-art fusion methods on data from miscalibrated sensors and achieves similar or better results when it comes to LiDAR layer removal and point cloud reduction. We also provide the code for our benchmarking experiments, so that others can reproduce our results as well as test their methods on corrupted sensor data, making multimodal fusion more reliable in real-world scenarios https://github.com/ViktorKare/bevf.

II. Sensor Fusion for 3D object detection

One of the main benefits of the RGB camera in an object detection scene is the semantic-rich nature of the data. Each image holds hundreds of thousands of pixels that are closely semantically related. LiDAR data on the other hand does not carry semantic information and even expensive high-resolution

Manuscript received: June 12, 2023; Revised July 31, 2023; Accepted August 22, 2023

This paper was recommended for publication by Editor Pascal Vasseur upon evaluation of the Associate Editor and Reviewers

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation

*Both authors contributed equally to this work

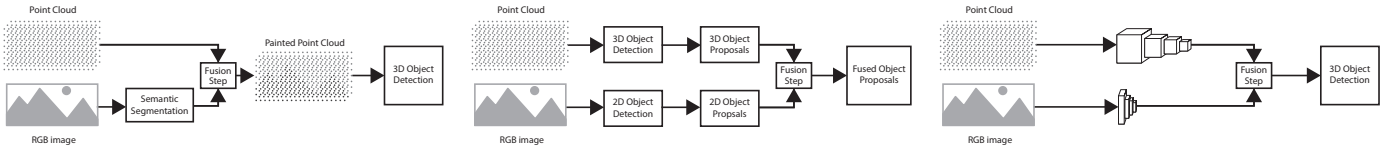
[1]Maciej K. Wozniak, Viktor Kårefjärds, and Patric Jensfelt are with the Division of Robotics, Perception, and Learning, KTH Royal Institute of Technology, Stockholm, Sweden maciejw@kth.se

[2]Marko Thiel is with the Institute for Technical Logistics, Hamburg University of Technology, Germany

Digital Object Identifier (DOI): see top of this page.

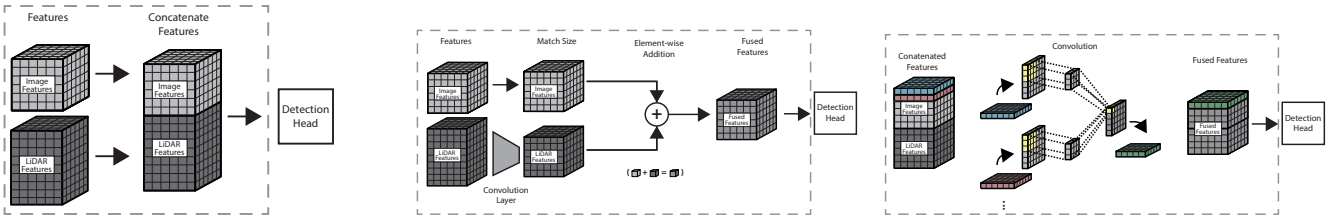
IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.



(a) Early fusion architecture. (b) Late fusion architecture. (c) Deep fusion architecture.

Fig. 1: Deep neural network fusion architectures.



(a) *Concatenation* fusion step. (b) *Element-wise addition* fusion step. (c) *Convolution* fusion step.

Fig. 2: Basic fusion steps. These methods are often used as an intermediate *stage* in other, more advanced fusion steps.

LiDAR sensors capture point clouds that are much sparser in comparison to RGB cameras, but their advantage is the ability to directly retain geometric information about the scene. This section describes *multimodal fusion* approaches used in 3D object detection that integrate camera and LiDAR to leverage the strengths of both sensors.

We describe common fusion architectures (where the fusion happens in the network) as well as the fusion steps (how the information is combined together) to support our analysis and design.

A. Multi-modal Fusion Architectures

Multi-modal 3D object detection models fuse data at various stages of the network [9], [10], [11], [4], [12]. The type of fusion can be categorized in accordance with the level at which the fusion is performed [13].

Early-fusion, showed in Fig. 1a, represents a data-level fusion, where the modalities are combined before any significant feature encoding. Strategies include a raw or processed image-to-point projection. Methods like [14], [10] fuse semantic segmentation of RGB image with the LiDAR point cloud. The main difficulty with this approach is the significant difference between the two data modalities at the early stage in the pipeline, which can make these methods prone to noise and corruption.

Late-fusion, showed in Fig. 1b, operates at the object level. In this type of approach, the image and LiDAR pipelines are largely isolated until proposals (e.g. bounding boxes) from each branch are generated [12]. Fusion here focuses on integrating proposals from two branches and incorporating features, such as confidence scores, to generate the model the final IoU score.

Deep-fusion, showed in Fig. 1c, performs fusion at the feature level where both modalities are first encoded by a neural network backbone (e.g. SwinTransformer for camera [15] and PointNet for LiDAR [16]) into the feature space and then combined together on the feature level. As Liu et al., Liang et al., or Bai et al. [11], [4], [6] showed, this approach is the most robust towards different disturbances in the data and performs significantly better than early or late fusion, however, it comes with an inference speed penalty.

B. Fusion step

There are many ways how the information can be fused together. We refer to this operation as a *fusion step* (marked in Fig. 1) and discuss it in detail in this section.

The simplest way to combine camera and point cloud feature tensors is through ***concatenation*** resulting in a large feature tensor, showed in Fig. 2a. This leaves the dense detection head with the unfused data, and consequently, the head learns to use the two modalities to perform detection. The potential strength of this fusion step approach is the low information loss. With a sophisticated detection head, the choice to keep the two feature spaces separated could be an advantage. This approach is used as an intermediate operation in most of the methods, however, PointPainting [17] uses it as a main part of the fusion-step block.

The ***element-wise addition*** fusion step is feature-to-feature addition, where each feature value in the LiDAR tensor is added with the respective value in the image feature tensor to create fused features of the same size, see Fig. 2b. Note how the feature tensors from the two data streams must be the same size to perform the step. This fusion step can be found e.g. in MVXNet [10], although no point-wise operations are performed in this fusion step as a consequence of the voxelized feature space.

The next methods use different types of neural network layers to fuse the signals. The ***convolution*** fusion step operates on a concatenated tensor originating from the two separated data sources. Once concatenated, the fusion step is not different from a standard channel-reducing convolution. A kernel (sliding window) operates on the feature space by sliding over the larger concatenated feature tensor (see Fig. 2c). The operation is then repeated to include all combinations. Notice how the number of kernels is selected in such a way that the resulting fused feature tensors are reduced along the channel dimension.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

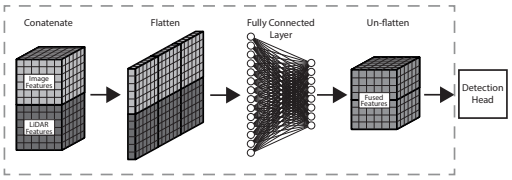


Fig. 3: *Fully connected* fusion step.

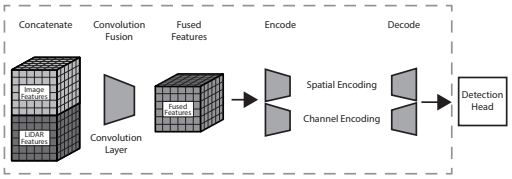


Fig. 4: *Encoder and decoder* fusion step.

The *convolution* is used in many methods as an intermediate or standalone fusion step in many SOTA methods [11], [4].

We can think about previous fusion steps, showed in Fig. 2, as basic building blocks. The following, more advanced approaches, often use them as intermediate operations.

Fully connected (showed in Fig. 3) starts from concatenating two feature tensors from the image and point cloud into one spatial dimension. Next, the tensor is flattened along the other spatial dimension and used as input for the fully connected layer. This fusion module includes batch normalization and an activation function after the fully connected layer, before feeding this information into the detection head. PointFusion uses a similar approach with multiple fully connected layers (MLP) [18]).

Zhijian et al. [11] proposed the ***encoder-decoder*** fusion step to address spatial and channel misalignment. We refer to *channel misalignment* as a misalignment between features in the LiDAR and camera feature spaces in the channel direction. As showed in Fig. 4 a small encoder-decoder network module was added after the convolution fusion step.

First, channel-wise encoding is used where the channel space is encoded to a smaller space, and then this is decoded to upscale it to the original channel size. The idea behind this step is to target any channel-wise misalignment.

The second parallel step is spatial encoding and decoding where the spatial dimensions are encoded to a smaller space and in a similar way, a decoder is applied in succession to restore the original dimensions. This feature aligning encoding-decoding is applied to account for misalignment but it also comes with more learnable parameters.

Additionally, two-way encoding does not have any non-encoded information pass through (skip connection) that helps perceive non-encoded information and does not share the information between the channels.

Hu et al. proposed the ***Squeeze-and-Excitation*** (SE) [19], that was later used as a fusion step by Liang et al. [4], showed in Fig. 5.

The *squeeze* step makes a global average pooling to aggregate features in order to create channel-wise descriptors. The *excitation* step then uses fully connected layers to produce channel-wise activations that are applied to the map of features.

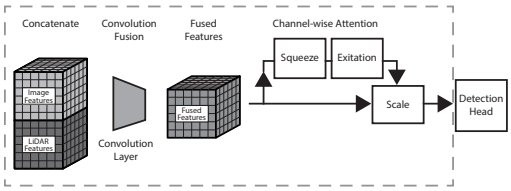


Fig. 5: *SE block* fusion step.

The output of the SE block can then be used to scale the convolutional features, according to information value. Thus, essential interdependencies can be enhanced.

It is important to mention that the *fusion steps* described above were used in many multimodal fusion methods as standalone steps or as one of the intermediate processes in the fusion step. There are also ***other*** architectures worth mentioning, such as the transformer-based fusion step in Transfusion [6] or probability voting step in CLOCs [12]. Moreover, some methods use a step architecture that resembles some of the fusion steps but are hard to classify as one category, such as RoIFusion [20]. For the sake of this research, we focus on the ones described in depth in this section, since they are used in the best performing methods.

III. Methods

Fusion steps presented in Section II often struggle to maintain the same level of performance on corrupted data such as sensor misalignment, lower resolution, or missing points, as they do on correct data. In light of this, we propose a novel fusion step that enhances the robustness and reliability of the fusion process, even in the presence of corrupted data.

Our fusion step, showed in Fig. 6, draws the inspiration from previously developed fusion steps enhancing their robustness by alternating them and combining them together, leading to improved overall performance.

The fusion step starts with a convolution, followed by an encoder-decoder structure and a Squeeze-and-Excitation (SE) block. The encoder architecture contains three branches that work in parallel. In the first branch, the fusion features are *passed through* (skip-connection) through a one-layer encoding step. The second *spatial encoding* branch reduces the spatial dimension and upscales to the appropriate dimensions after the decoder layers. The original 200×200 feature space is reduced to 100×100 and back again. The third branch performs the same spatial encoding, but also, a channel encoding, where the channel space is funneled to half the original size.

Next, the SE block attentively scales the relationships between the fused feature channels. The *squeeze* and *excitation* operations enhance the important information by modeling the channel inter-dependencies through an adaptive average pooling operation, establishing and enhancing the relationships between channels as they might otherwise be lost in the channel-reducing convolution fusion process.

As we emphasized before, our method was carefully designed to address the issues caused by lower sensor resolution, missing LiDAR data or sensor misalignment, but we also want to ensure that our method performs on par with other SOTA

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

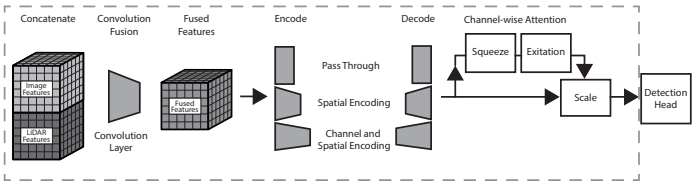


Fig. 6: **Our method**: the convolution fusion step, followed by encoder-decoder with SE-block.

methods when data is not corrupted. Therefore, in our encoded-decoder block, the first branch is an information pass-through that allows the model to operate with high performance in a non-misaligned scenario. The other two branches account for sensor-to-sensor misalignment.

The *spatial encoding branch* facilitates the association of spatial-neighboring features from the two data streams, as they are likely to represent the same object with slight spatial misalignment. The *channel and spatial encoding* branch includes the encoding which is added to account for misalignment in the channel space (between features in the LiDAR and camera feature spaces in the channel direction) since object features can be represented differently in the two feature modalities.

While we tried different numbers of encoding branches, such as just spatial encoding, we got the best results for corrupted and uncorrupted cases with a three-branch encoder-decoder.

At this stage, the input to the *SE-block* includes features of the same scene encoded in three different ways by each of the encoder-decoder branches. The *SE-block* attentively associates the combined three-branch feature spaces. The multi-scale features include to a degree the same object three times, one from each of the three branches. The following *SE-block* is thus of significant importance as it associates those channels together after the encoder-decoder, ahead of the detection head.

IV. Experimental Setup

This section describes metrics and the experimental setup used for evaluation. Through the experiments, we examine real-world scenarios often occurring in the context of robotics platforms, when understanding the environment is hindered due to partial sensor failure, sensor misalignment, or lower sensor resolution.

Our experiments are divided into two main parts. In the first part, we test the robustness of different SOTA methods for 3D object detection and see what performance decrease we can expect to choose the most robust method. We focus on fusion methods, but also test camera and LiDAR-only methods for reference. In the second part, we evaluate our proposed fusion step when there is sensor misalignment and with the best performing method from the first part as a baseline by replacing their fusion step with ours.

We run the experiments on a desktop computer with Intel Core i7-12700KF (12 cores, 20 threads) 5.00 GHz CPU and an NVIDIA GeForce RTX 3090 (24 GB) GPU.

A. Metrics

Average Precision (*AP*) is defined as the area under the precision-recall curve, $P(r)$, where *mAP* is the class-wise mean of *AP*. Here, N is the number of classes, and AP_k is the average precision for class k, see Eq. (1).

$$AP = \int P(r)dr \quad , \; mAP = \frac{1}{N}\sum_{k=1}^{N} AP_k \tag{1}$$

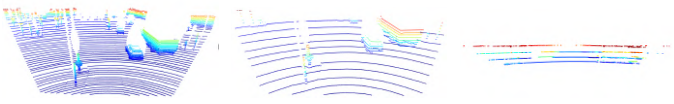
The *KITTI* [21] metrics require a 70% Intersection over Union (*IoU*) for cars (moderate difficulty) with a minimum bounding box height of 25 pixels and a maximum truncation of 30%. The *nuScenes* [22] deviates from the definition of a match. It is determined by thresholding the 2D center distance on the ground plane instead of using the *IoU*. This results in *mAP* score being up to 2× higher on *KITTI* than *nuScenes* for the same methods, thus the numerical values cannot be directly compared between the Table I and Table II. The mean Average Precision *mAP* is then calculated by threshold averaging, based on distance $\mathbb{D} = \{0.5, 1, 2, 4\}$ in meter. We also used nuScenes Detection Score *NDS* Eq. (2), here $\mathbb{TP}$ is a set of five true positive metrics described in detail in [22].

$$NDS = \frac{1}{10}(5\ mAP + \sum_{mTP \in \mathbb{TP}} (1 - min(1,\ mTP))) \tag{2}$$

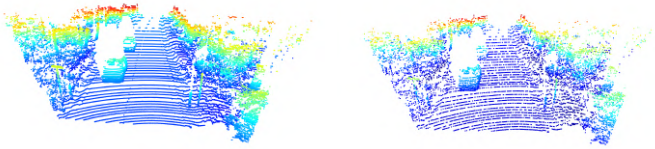
B. LiDAR data corruption

Different datasets, smaller mobile robots, or otherwise less expensive autonomous driving setups commonly make use of lower-resolution LiDAR sensors, as in Fig. 7b or Fig. 7c. Thus, we simulate and test how reducing LiDAR resolution to 16, 4, and 1 layer impacts the 3D object detection model and fusion steps.

Additionally, a wide variety of disturbances can impact the quality of measurement from a LiDAR sensor. A generic and overarching strategy to highlight how sensitive the multi-modal



(a) Original LiDAR point cloud. (b) LiDAR point cloud reduced to 16 layers. (c) LiDAR point cloud reduction to 4 layers.

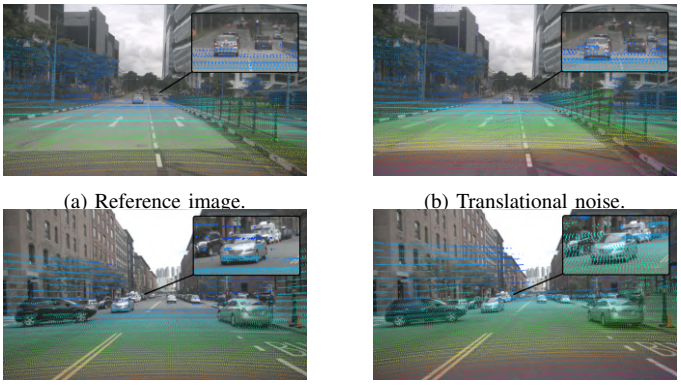


(d) Reference point cloud (e) 50% point reduction.

Fig. 7: Example of LiDAR layer and points reduction.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.



(a) Reference image. (b) Translational noise.

(c) Reference image. (d) Rotational noise.

Fig. 8: Visualization of sensors misalignment problem.

object detection methods are to the absence of high-quality LiDAR data is to simulate point cloud density reduction. In cases of low-reflection, due to snow, rain, or other environmental effects, the LiDAR reflection beams can be lost and, thus, the point cloud consists of fewer points Fig. 7e. We simulate this scenario by removing the points on seeded pseudo-random sampling at different ratios of point dropping.

C. Camera-LiDAR missaglingment

To further test the fusion steps with respect to the multi-sensor misalignment problem and misalignment as a result of poor calibration, we propose a misalignment experiment in which the fusion steps will be subject to purposefully added misalignment between the camera and the LiDAR sensor.

In addition to the tests proposed by [6], which only handles translation misalignment, our experiments also include rotational misalignment and the combination of both, a commonly occurring problem in the robotics field. This is achieved by adding random noise to the transfer matrices between the respective camera frames, and the joint reference frame, resulting in a shift between the point cloud and camera, as showed in Fig. 8a and Fig. 8b. Thus, in the joint feature space, the soon-to-be-fused features are spatially misaligned, and the fusion step must be performed in a way that respects any such misalignment.

The experiment is performed on a series of translation and rotation misalignments. In the translation case, noise is added to all three directions simultaneously, x, y, z. In the rotational experiment, noise is added at the same time to $roll, pitch$, and yaw.

Note, that results for uncorrupted data in Table IV are slightly different than in Table I and Table III due to the different train/test split.

V. Experiments

Our experiments are divided into two main parts. In the first part, we test the robustness of different SOTA methods for 3D object detection and see what performance decrease we can expect to choose the most robust method.

In the second part, we use the most robust model and test SOTA and our fusion steps against sensor misalignment.

TABLE I: Result from the LiDAR layer removal and LiDAR point reduction on nuScenes [22].

			Scores on nuScenes [22]		
Method	**Defect**		mAP	NDS	ΔmAP
BEVFusion-Liang [4]	*Layer*	32	54.01	60.66	
with PointPillars [2] configuration	*removal*	16	47.52	56.66	-12.0%
Fusion type: *Deep-fusion*		4	42.10	52.80	-22.1%
Fusion step: *Convolution with*		1	15.23	34.75	-71.8%
SE-block	*Points*	100%	54.01	60.66	
	reduction	90%	53.83	60.53	-0.3%
		80%	53.58	60.40	-0.8%
		50%	51.22	58.87	-5.2%
TransFusion [6]	*Layer*	32	58.95	54.27	
Fusion type: *Deep-fusion*	*removal*	16	42.40	45.24	-28.1%
Fusion step: *Transformers based*		4	27.06	36.43	-54.1%
on object queries		1	02.22	11.54	-96.2%
	Points	100%	58.95	54.27	
	reduction	90%	58.48	54.05	-0.9%
		80%	57.83	53.69	-1.8%
		50%	53.79	51.51	-8.7%
PointPillars [2]	*Layer*	32	39.71	53.15	
Single modal: LiDAR only	*removal*	16	28.64	46.59	-27.9%
		4	15.61	38.35	-60.7%
		1	0.64	11.61	-98.4%
	Points	100%	39.71	53.15	
	reduction	90%	39.40	52.98	-0.8%
		80%	39.03	52.72	-1.9%
		50%	36.39	51.12	-8.8%
FCOS3D [3]	*Layer removal*	32, 16, 4, 1	29.80	37.74	-
Single modal:	*Points reduction*	100%, 90%,	29.80	37.74	-
RGB camera only (no affected by Lidar)		80%, 50%			

[1] ΔmAP indicate the percentage decrease in mAP in relation to the non-reduced baseline

TABLE II: Result from the LiDAR layer removal and LiDAR point reduction on KITTI [21].

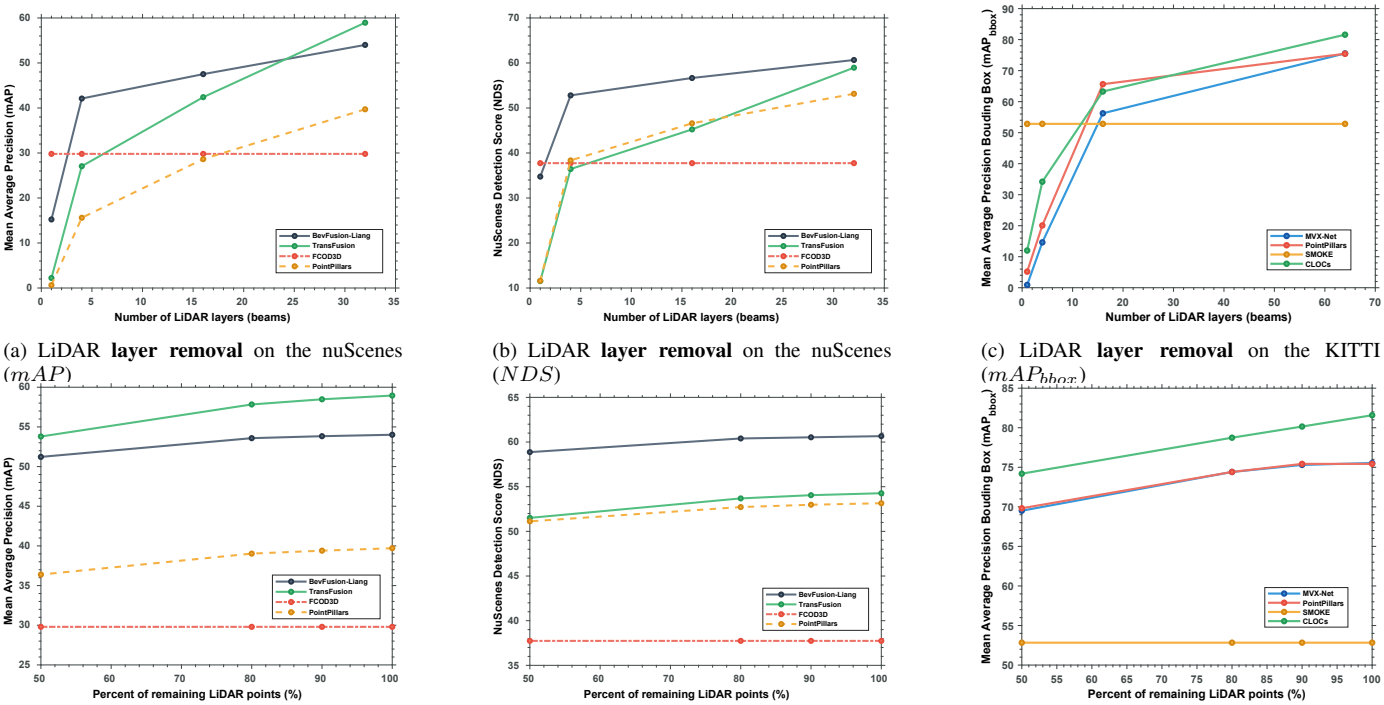
			Scores on KITTI [21]		
Method	**Defect**		mAP_{3D}	mAP_{bbox}	ΔmAP_{bbox}
MVX-Net [10]	*Layer*	64	62.92	75.54	
Fusion type: *Early-fusion*	*removal*	16	43.48	56.23	-30.9%
Fusion step: *Point-wise*		4	6.04	14.62	-90.4%
concatenate		1	0.02	0.90	-99.9%
	Points	100%	62.92	75.54	
	reduction	90%	62.37	75.30	-0.8%
		80%	61.48	74.41	-2.3%
		50%	56.38	69.51	-10.4%
CLOCs [12]	*Layer*	64	69.02	81.58	
Fusion type: *Late-fusion*	*removal*	16	46.51	63.26	-32.6%
Fusion step: *Object*		4	8.04	34.18	-88.4%
candidate probability scoring		1	1.98	11.99	-97.1%
Note: *Calculated from*	*Points*	100%	69.02	81.58	
class-specific models	*reduction*	90%	67.88	80.15	-1.7%
		80%	66.96	78.74	-3.0%
		50%	60.30	74.19	-12.6%
PointPillars [2]	*Layer*	64	64.36	75.43	
Single modal: *LiDAR only*	*removal*	16	47.96	65.63	-25.5%
		4	15.22	20.07	-76.4%
		1	0.90	5.19	-98.6%
	Points	100%	64.36	75.43	
	reduction	90%	63.63	75.43	-1.1%
		80%	62.11	74.51	-3.5%
		50%	55.73	69.82	-13.4%
SMOKE [23]	*Layer removal*	64, 16, 4, 1	3.51	52.83	-
Single modal:	*Points reduction*	100%, 90%,	3.51	52.83	-
RGB camera only		80%, 50%			
(not affected by LiDAR)					

A. Robustness experiments - layer and LiDAR point removal

In the first two experiments, we evaluate SOTA methods on lower density LiDAR data. In Table I and Table II the $fusion\ type$ section denotes at what level the respective method fuses the data streams using the taxonomy as introduced in Section II. The $fusion\ step$ section denotes how that fusion is realized.

In the LiDAR layer removal experiment, each method is evaluated on $16, 4$, and 1 layered point cloud data on 64 (*KITTI*) or 32 (*nuScenes*). The results are showed in

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.



(a) LiDAR **layer removal** on the nuScenes (mAP)

(b) LiDAR **layer removal** on the nuScenes (NDS)

(c) LiDAR **layer removal** on the KITTI (mAP_{bbox})

(d) LiDAR **point number reduction** on the nuScenes (mAP)

(e) LiDAR **point number reduction** on the nuScenes (NDS)

(f) LiDAR **point number reduction** on the KITTI (mAP_{bbox})

Fig. 9: Comparison of different methods' performance when tested on data with reduced LiDAR layers and reduced point cloud.

TABLE III: LiDAR layer and point number reduction on our fusion step versus baseline fusion step.

			Scores on nuScenes [22]		
Metod	**Defect**		mAP	NDS	ΔmAP
Our fusion step: *Convolution with encoder-decoder SE-block Augmented*	*Layer removal*	32	51.69	56.18	
		16	46.38	54.22	-10.2%
		4	**42.65**	51.94	-17.5%
		1	**20.98**	37.90	-59.4%
	Points reduction	100%	51.69	56.18	
		90%	51.65	57.45	-0.01%
		80%	51.57	57.45	-0.2%
		50%	49.68	56.32	-3.9%
Beseline fusion step *Convolution with SE-block* [4]	*Layer removal*	32	54.01	60.66	
		16	**47.52**	56.66	-12.0%
		4	42.10	52.80	-22.1%
		1	15.23	34.75	-71.8%
	Points reduction	100%	54.01	60.66	
		90%	**53.83**	60.53	-0.3%
		80%	**53.58**	60.40	-0.8%
		50%	**51.22**	58.87	-5.2%

Table I and Table II and Fig. 9, highlight how the early-fusion method, MVX-Net [10] and the late-fusion method, CLOCS [12] show significant performance drops as the layer number decreased, in comparison to much more sable deep-fusion methods Transfusion [6] and BEVFusion-Liang [4]. The single-modal LiDAR-only PointPillars [2] largely follows the decrease in performance. This highlights the fact that presented fusion methods still heavily depend on high-resolution LiDAR data and largely fail to operate independently on the unaffected RGB images when LiDAR data is corrupted.

In the point cloud reduction experiment, we evaluate each method on point clouds randomly reduced to $90\%, 80\%$, and 50% to simulate LiDAR performance deviations. We can once again observe how the BEVFusion-Liang [4] can retain performance as the LiDAR point cloud data is affected negatively. Further, the early fusion method MVX-Net [10] outperforms the LiDAR-only PointPillars [2] model in the most extreme case. In this case, the point cloud is affected in an unordered way and fusion methods find a way to associate corresponding point clouds with the image feature, resulting in a less significant performance drop.

When we compare both deep fusion methods, we can see that BEVFusion-Liang [4] outperforms Transfusion [6] on LiDAR layer removal and is slightly worse when it comes to point cloud reduction. Nevertheless, the percentage-wise decrease in performance is much smaller, suggesting it is more robust to various data perturbations.

B. Proposed fusion step

The results above led us to choose BEVFusion-Liang [4] as the baseline for the evaluation of our proposed fusion step.

We benchmark the baseline against a version with our fusion step Section III on the experiments in Section V-A. The results in Table III highlight how our fusion step performs very similarly to the baseline BEVFusion [4] model when it comes to the LiDAR layer removal and point cloud reduction. While our method achieves higher mAP for most LiDAR layer removal scenarios, it performs only slightly worse compared to the baseline on point cloud reduction.

We can also observe that the percentage-wise drop for our fusion step is lower than the baseline, proving that our solution is more robust and less prone to data corruption. The key reason behind the comparatively lower percentage-wise performance decrease lies in our fusion step architecture. Our fusion step proves to be reliable while handling corrupted

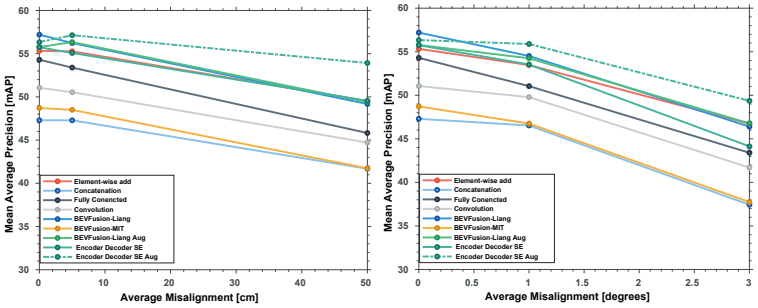
IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

TABLE IV: 3D object detection results from sensor misalignment with different fusion steps experiments on the chosen model [11].

		Misalignment with max. limits. nuScenes [22]						
Fusion Step	**Metric**	**None**	**10cm**	**100cm**	**1°**	**3°**	**10cm ∪ 1°**	**#params**
Element-wise add	mAP	55.33	55.27	49.52	53.47	46.73	53.37	
as in MVXNet[10]	NDS	55.17	52.95	48.23	51.39	45.57	51.47	88.0M
Concatenation	mAP	47.30	47.29	41.69	46.53	37.43	39.43	
as in PointPainting[17]	NDS	45.34	45.70	41.55	44.80	38.36	40.32	87.2M
Fully connected	mAP	54.30	53.39	45.82	51.05	43.40	49.97	
as in PointFusion[18]	NDS	53.87	51.18	46.19	49.53	44.22	49.17	87.3M
Convolution	mAP	51.06	50.54	44.71	49.79	41.70	44.69	
as an option in [4] or [11]	NDS	49.68	48.99	44.79	48.34	42.79	43.91	89.3M
Convolution with encoder-decoder	mAP	48.72	48.50	41.73	46.75	37.74	47.20	
as in BEVFusion-MIT [11]	NDS	49.93	48.48	43.90	46.75	41.15	47.34	93.1M
Convolution with SE-block (**Baseline**)	mAP	**57.20**	56.23	49.18	54.51	46.40	54.08	
as in BEVFusion-Liang [4]	NDS	56.65	53.34	48.40	52.15	46.10	51.74	89.5M
Convolution with SE-block	mAP	55.77	56.34	49.40	54.25	46.77	53.47	
Baseline *Augmented as in BEVFusion-Liang [4]*	NDS	55.46	53.36	48.58	52.03	46.94	51.58	89.5M
Convolution with encoder	mAP	55.75	55.08	49.55	53.53	44.13	52.25	
decoder and SE-block (Ours)	NDS	55.24	52.32	49.00	51.68	46.11	50.59	95.0M
Convolution with encoder	mAP	56.34	**57.14**	**53.92**	**55.89**	**49.36**	**56.15**	
decoder and SE-block *Augmented* (**Ours**)	NDS	**57.02**	**54.85**	**53.31**	**54.44**	**51.56**	**54.33**	95.0M

[1]The best results are in **bold** and second-best are underlined.



(a) Sensor misalignment – translation (b) Sensor misalignment – rotation

Fig. 10: Robustness evaluation of fusion steps on sensor misalignment.

data and dealing with unexpected disturbances. This added robustness enables our fusion step to maintain more stable performance even when parts of the LiDAR layers or points are removed.

C. Proposed fusion step - sensor misalignment

Sensors are often miscalibrated or decalibrate during the robot's movement. Fusion methods must account for these issues to be useful in real-world applications. The results of the sensor misalignment experiments are presented in Table IV and Fig. 10. During the training, the whole network (and fusion step) is trained for 6 epochs in addition to the LiDAR backbone which is pretrained for 24 epochs, and the image backbone is pretrained for 36 epochs. We trained and tested all the fusion steps on the *nuScenes* data set. The training schedule uses an ADAM optimizer and step-downs learning rate at epoch 4 and epoch 6, lowered to 10^{-4}, and then to 10^{-5} in the final epoch.

The convolution with encoder-decoder and SE-block marked with augmentation have the data abstraction applied during the training. This pipeline is *identical* to the step-down learning schedule, except for the last epoch where a small amount of noise, randomly chosen data corruption - small sensor misalignment or removal/addition of a small number of points to the point cloud, is added. The idea is that the noise makes the fusion step more general and less susceptible to data corruption. The results of the misalignment experiments give insight into the performance of each fusion step in normal conditions, light, and finally severe misalignment.

While our method benefits from this approach the other methods did not, and their performance dropped by $5-10\%$ in most of the cases. For that reason, in the Table IV we only included *augmented* results for our method and the baseline.

The results showed in Table IV highlight how the concatenation is at large the worst performer of the three simple fusion steps: convolution, element-wise add, and concatenation. We can also observe how the element-wise add is one of the best-performing fusion steps overall, despite its simple nature. Further, the convolution with SE-block is the best performer in the non-misaligned case, but our method, convolution with encoder-decoder and SE-block, outperforms the other methods in small and large translational and rotational misalignments.

Our augmented method performs better than the non-augmented version in all testing scenarios. Thus, we conclude that adding noise not only assists misaligned cases but also generalizes the fusion operation at large, achieving the best performance in almost every case on corrupted and uncorrupted data.

To summarize, our approach surpasses both the baseline and other state-of-the-art fusion step techniques when evaluated for sensor misalignment, exhibiting the lowest performance drop in both easy and hard cases.

D. Model Size and inference speed

In Table IV we showed the number of parameters required by each model. The similarity between them comes from the

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

fact that the number of all the parameters does not change much (percentage-wise) between different fusion steps, since the fusion step block represents only a small part of all the model's parameters (models tested in Table 4 have from 87 to 95M of parameters) and the image backbone SwinTransformer [15] is *responsible* for 55.7M parameters.

We have identified that the creation of the BEV, which is performed by the LSS network [24], is the slowest part of our system. That resulted in an inference speed of 0.8 FPS regardless of the fusion step we used.

Finally, we also tested a smaller version of our model (62.3M parameters) using ResNet-50 image backbone with 23M parameters. Regardless of the much smaller backbone and overall model, we still got a very slow inference time (between 1.1-1.3 FPS), since the LSS network is the main bottleneck. Additionally, this method performed much worse on the baseline uncorrupted data ($mAP = 16.07$ and $NDS = 24.22$), thus we did not run any further experiments with it.

VI. Discussion and future work

In this work, we have developed a novel fusion step for 3D object detection, robust and adaptable to various real-world scenarios. We compared our fusion step against a wide range of existing fusion methods. Our approach consistently outperformed SOTA methods across different scenarios, particularly where sensors are misaligned, and demonstrated greater robustness (lower percentage-wise decrease) when tested on LiDAR layer removal and point reduction. Additionally, it is comparable to other SOTA methods when evaluated on uncorrupted data.

Although our proposed fusion step demonstrates good performance, the improvement is limited due to the impact of sensor misalignment and lower resolution of the LiDAR on far away objects, compared to those nearby. In certain applications, such as mobile robots using lower-resolution LiDAR sensors, the solution could involve restricting LiDAR maximum distance during training and testing the methods since in these applications it is more crucial to detect nearby objects than ones $50 - 70$m away.

Finally, it would be beneficial to address challenging weather conditions, such as rain or fog, that can significantly degrade the quality of both sensor inputs. These questions will be deferred to future research, as they require further investigation and analysis.

References

[1] A. Khoche, M. K. Wozniak, D. Duberg, and P. Jensfelt, "Semantic 3d grid maps for autonomous driving," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2681–2688, IEEE, 2022.

[2] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12697–12705, 2019.

[3] T. Wang, X. Zhu, J. Pang, and D. 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.

[4] T. Liang, H. Xie, K. Yu, Z. Xia, Z. Lin, Y. Wang, T. Tang, B. Wang, and Z. Tang, "Bevfusion: A simple and robust lidar-camera fusion framework," *arXiv preprint arXiv:2205.13790*, 2022.

[5] S. Das, L. af Klinteberg, M. Fallon, and S. Chatterjee, "Observability-aware online multi-lidar extrinsic calibration," *IEEE Robotics and Automation Letters*, vol. 8, no. 5, pp. 2860–2867, 2023.

[6] X. Bai, Z. Hu, X. Zhu, Q. Huang, Y. Chen, H. Fu, and C.-L. Tai, "Transfusion: Robust lidar-camera fusion for 3d object detection with transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1090–1099, 2022.

[7] T.-M. Nguyen, S. Yuan, M. Cao, Y. Lyu, T. H. Nguyen, and L. Xie, "Ntu viral: A visual-inertial-ranging-lidar dataset, from an aerial vehicle viewpoint," *The International Journal of Robotics Research*, vol. 41, no. 3, pp. 270–280, 2022.

[8] M. K. Wozniak, V. Kårefjärd, M. Hansson, M. Thiel, and P. Jensfelt, "Applying 3d object detection from self-driving cars to mobile robots: A survey and experiments," in *2023 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pp. 3–9, IEEE, 2023.

[9] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1907–1915, 2017.

[10] V. A. Sindagi, Y. Zhou, and O. Tuzel, "Mvx-net: Multimodal voxelnet for 3d object detection," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 7276–7282, IEEE, 2019.

[11] Z. Liu, H. Tang, A. Amini, X. Yang, H. Mao, D. Rus, and S. Han, "Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation," *arXiv preprint arXiv:2205.13542*, 2022.

[12] S. Pang, D. Morris, and H. Radha, "Clocs: Camera-lidar object candidates fusion for 3d object detection," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 10386–10393, IEEE, 2020.

[13] K. Huang, B. Shi, X. Li, X. Li, S. Huang, and Y. Li, "Multi-modal sensor fusion for auto driving perception: A survey," *arXiv preprint arXiv:2202.02703*, 2022.

[14] S. Vora, A. H. Lang, B. Helou, and O. 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.

[15] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021.

[16] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

[17] S. Vora, A. H. Lang, B. Helou, and O. 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.

[18] D. Xu, D. Anguelov, and A. Jain, "Pointfusion: Deep sensor fusion for 3d bounding box estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 244–253, 2018.

[19] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

[20] C. Chen, L. Z. Fragonara, and A. Tsourdos, "Roifusion: 3d object detection from lidar and vision," *IEEE Access*, vol. 9, pp. 51710–51721, 2021.

[21] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[22] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," *arXiv preprint arXiv:1903.11027*, 2019.

[23] Z. Liu, Z. Wu, and R. Tóth, "Smoke: Single-stage monocular 3d object detection via keypoint estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 996–997, 2020.

[24] J. Philion and S. Fidler, "Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*, pp. 194–210, Springer, 2020.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.