Lift-Attend-Splat: Bird's-eye-view camera-lidar fusion using transformers

Supplementary Material

A. Monocular depth in the "LiftSplat" paradigm

A.1. Computation of ground truth depth from lidar

For each camera image we compute the ground depth map $D^{\text{gt}} \in \mathbb{R}^{H \times W}$ by projecting the 3D lidar point cloud onto the image plane and binning each point within the pixels of the camera feature map. For non-empty cells, we follow [27] and choose the depth to be the minimum distance (from the camera plane) of all the points in the cell, leaving the depth unspecified for empty cells and those for which the minimum yields a depth value which is outside the range of the model's depth bins. This depth map is suitable for visualisation and depth metric evaluation, but for depth supervision it is necessary to calculate the one-hot encoding of D^{gt} according to buckets defined by the model's depth bins $d \in \mathbb{R}^{N_D}$.

A.2. Visualisation of depth maps

We generate the monocular depth maps shown in Fig. 1 by calculating the weighted average of the model's depth bins d with the predicted depth distribution $D^{\text{pred}} \in \mathbb{R}^{N_D \times H \times W}$

$$D^{\text{mean}}_{h,w} = \sum_{n}^{N_D} d_n D^{\text{pred}}_{n,h,w}. \tag{6}$$

This depth map is constrained by construction to $[\min(d), \max(d)]$ and we map this range onto the Turbo colour map [38] for visualisation.

The lidar depth map D^{gt} is similarly colourised, except for cells where the depth is unspecified as described above (see Fig. S1, top-right) which are coloured grey.

A.3. Supervision of predicted depth using lidar

We perform all of our experiments using the method presented in [36] and use the original repository[3]. We use the vanilla Lift-Splat transform implemented in the class `LSSTransform` with default parameters provided in the original work. We supervise the depth classifier by introducing the following loss alongside the original detection losses,

$$L_{\text{depth}} = -\frac{1}{N} \sum_{n}^{N} \log(D_n \cdot 1_n), \tag{7}$$

which is a cross-entropy loss between the lidar depth distribution and predicted depth distribution, taken over all cells for which the lidar depth is available. $D_n \in \mathbb{R}^{N_D}$ is the normalised predicted depth distribution from the LiftSplat model for the nth cell, 1_n is the one-hot encoded lidar depth distribution for the nth cell. The model is trained end-to-end with all components unfrozen as in [36] and hyper-parameter λ controlling the strength of the depth supervision loss with respect to the detection losses.

We also experiment with pretraining the depth estimation module within LiftSplat. First, we train the camera stream in [36] supervising only the depth distribution with the whole camera pipeline unfrozen. Following this pretraining, we add the lidar components and train the full model end-to-end as in [36], with no depth supervision ($\lambda = 0$) and all modules unfrozen.

[3]https://github.com/mit-han-lab/bevfusion

A.4. Extended depth quality results

We evaluate the performance of the depth classifier using five of the metrics proposed in [10]: root mean squared error (RMSE), root mean squared logarithmic error (RMSLE), mean absolute relative error (Abs. Rel.), mean squared relative error (Sq. Rel.) and fraction outside 125% (Frac. 125). We show all the metrics for 2 different methods of translating the classification output into a depth map: "mode" — in which we use the bin with maximum probability, and "mean" — where the predicted depth is the weighted average of all the bins. We take the average of these quantities over all predictions made by the depth classifier for which we have ground truth to compute the metrics. Camera feature cells for which lidar depth is unspecified are ignored. We compare BEVFusion and four different variants: adding depth supervision using Eq. (2) with various weights λ, using lidar depth maps instead of monocular depth estimation (lidar), using a pretrained and frozen depth classifier (pretrained), and finally removing depth estimation altogether by projecting camera features at all depths uniformly using Eq. (3) (uniform depth). Quantitative results can be seen in Tab. S1 and qualitative comparisons in Fig. S1.

B. Detailed experimental results

B.1. 3D object detection

In Tab. S2 we present per-class detection scores and compare our model to other state-of-the-art models on the validation and test splits of the nuScenes dataset. Our method outperforms baselines based on the LiftSplat paradigm. We are additionally showing how test-time-augmentations and temporal feature aggregation further improves these results.

Loss Weight	mAP	mode					mean				
		Relative		RMSE		Frac. 125	Relative		RMSE		Frac. 125
		Abs.	Sq.	Linear	Log		Abs.	Sq.	Linear	Log	
BEVFusion [36]	**68.5**	2.95	133.76	25.95	1.87	0.97	2.75	61.31	17.40	1.30	0.88
$\lambda = 0$	68.4	3.69	176.09	30.22	1.90	0.96	2.83	68.73	18.54	1.34	0.87
0.001	68.1	1.79	65.63	20.16	1.77	0.94	3.14	79.87	19.91	1.39	0.88
0.01	68.0	0.61	11.78	11.54	1.03	0.63	0.76	10.30	8.09	0.68	0.61
0.1	68.1	0.38	5.53	9.28	0.77	0.41	0.43	4.97	6.47	0.46	0.37
1	68.1	0.21	2.48	5.78	0.37	0.20	0.22	2.23	4.77	0.33	0.19
5	66.6	0.19	2.01	4.77	0.33	0.17	0.19	1.95	4.53	0.32	0.17
100	64.6	0.16	1.15	4.64	0.33	0.17	0.16	1.12	4.55	0.32	0.17
Pretrained	67.4	0.54	8.10	9.95	0.86	0.61	0.64	7.91	7.87	0.66	0.57
Lidar	68.4	0.04	0.01	0.29	0.05	0.00	0.04	0.01	0.29	0.05	0.00
Uniform depth	**68.5**	–	–	–	–	–	–	–	–	–	–

Table S1. Extended analysis of the monocular depth quality provided by different variations of the “LiftSplat” camera feature projection, see Sec. A.4.

Model	barrier	bicycle	bus	car	CV	MC	ped	TC	trailer	truck	mAP	NDS
Ours	74.1	70.0	81.3	90.3	33.8	80.8	89.3	79.7	44.0	68.2	71.2	72.7
BEVFusion[31]	73.5	67.5	77.7	89.1	30.9	79.0	89.4	79.3	42.6	66.7	69.6	72.1
DeepInteraction[66]	78.1	52.9	68.3	87.1	33.1	73.6	88.4	86.7	60.8	60.0	69.9	72.6
Ours‡	77.5	75.2	82.3	91.2	40.0	85.6	90.6	80.2	50.1	72.2	74.6	75.1
Ours w/ TFA	74.4	72.4	81.6	90.8	33.7	82.5	89.8	79.6	45.8	70.1	72.1	73.8
Ours‡ w/ TFA	78.6	78.2	84.3	91.6	39.9	87.5	91.4	80.7	51.2	73.3	75.7	76.0
Ours	78.0	54.9	72.1	89.0	38.9	75.3	90.3	87.0	65.3	64.2	71.5	73.6
Ours‡ w/ TFA	79.7	65.2	75.2	90.3	43.5	82.8	92.0	87.1	70.1	68.9	75.5	74.9
BEVFusion[31]	78.3	56.5	72.0	88.5	38.1	75.2	90.0	86.5	64.7	63.1	71.3	73.3
DeepInteraction[66]	80.4	54.5	70.8	87.9	37.5	75.4	91.7	87.2	63.8	60.2	70.8	73.4

Table S2. Per-class object detection scores on the nuScenes validation set (top) and test set (bottom). TFA: Temporal Feature Aggregation. ‡ indicates ensembling + TTA.

B.2. Detailed qualitative results

To obtain Figure 4a for our method, we first compute the full camera-to-BEV attention map $\text{Attn}^{(\text{cam}_i \rightarrow \text{bev})} \in \mathbb{R}^{H\times W\times N\times M}$. To do so, we extract the attention map of the last transformer decoder block by averaging over all heads, $\text{Attn}^{(\text{cam}_i \rightarrow \text{frustum})} \in \mathbb{R}^{H\times W\times D\times W'}$, where $(D \times W')$ corresponds to the frustum dimension. We construct $\text{Attn}^{(\text{cam}_i \rightarrow \text{bev})}$ by scattering the frustum attention values onto the BEV grid. Given camera image I_i, we then create $\text{Mask}^{(i)} \in \{0,1\}^{H\times W}$ by in-paint drawing the annotations, see Fig. S3, and obtain the BEV attention $\text{Attn}^{(\text{bev})} \in \mathbb{R}^{N\times M}$ shown on Figure 4a by projecting these camera features onto the BEV grid:

$$\text{Attn}^{(\text{bev})} = \max_{i,h,w} \text{Attn}^{(\text{cam}_i \rightarrow \text{bev})}_{h,w} \cdot \text{Mask}^{(i)}_{h,w}. \quad (8)$$

To obtain a similar visualisation for the “LiftSplat” projection, see Figure 4b, we adjust the implementation of [36] but use the same model weights. Firstly, we replace the feature map of image I_i with $\text{Mask}^{(i)}$ and use that as input to the projection. This binary mask is “lifted” onto a 3D point cloud using the normalised depth classification weights D_i for which we clipped the first 5 and last 5 depth bins. $\text{Mask}^{(i)}$ thus acts as an indicator function and D_i specifies the strength of correspondence between pixels and the 3D point cloud $P \in \mathbb{R}^{H\times W\times N_D\times 3}$. Secondly, during “splatting”, we project points onto the z = 0 plane and pool them using max. This operation ensures that the weight of attention for large objects in the final visualisation does not overpower that of smaller objects.

B.3. Ensemble and test-time augmentations

For test-time-augmentation (TTA) and model ensembling, we use WBF [50] based on L2 distance metric per object category to decide which of the boxes to fuse. We first carry out TTA (using mirror and rotation augmentations)

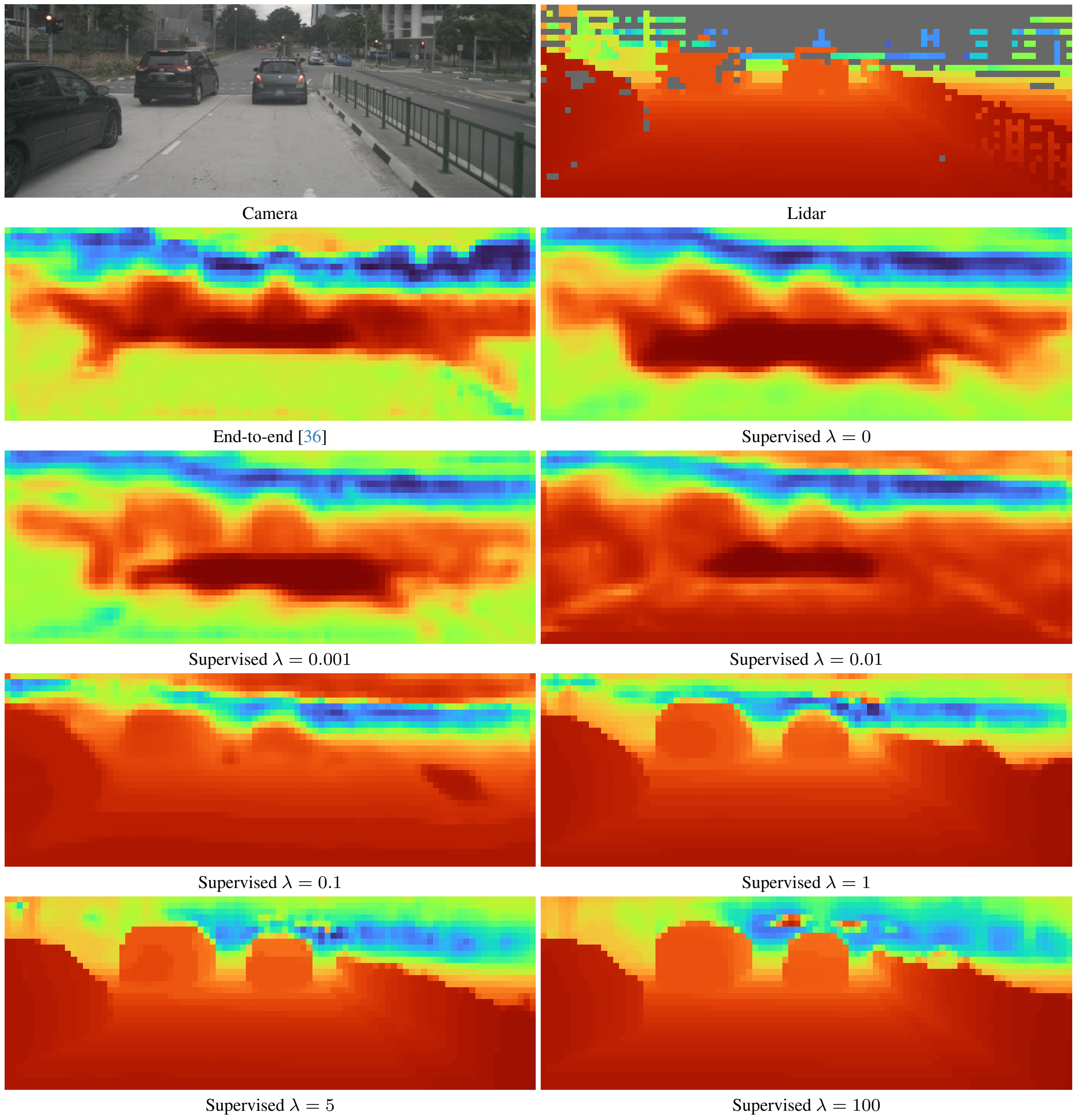
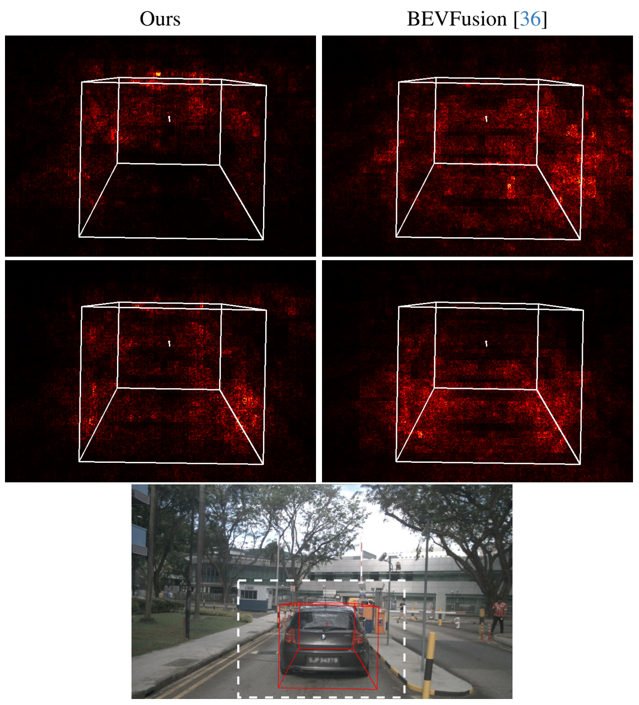
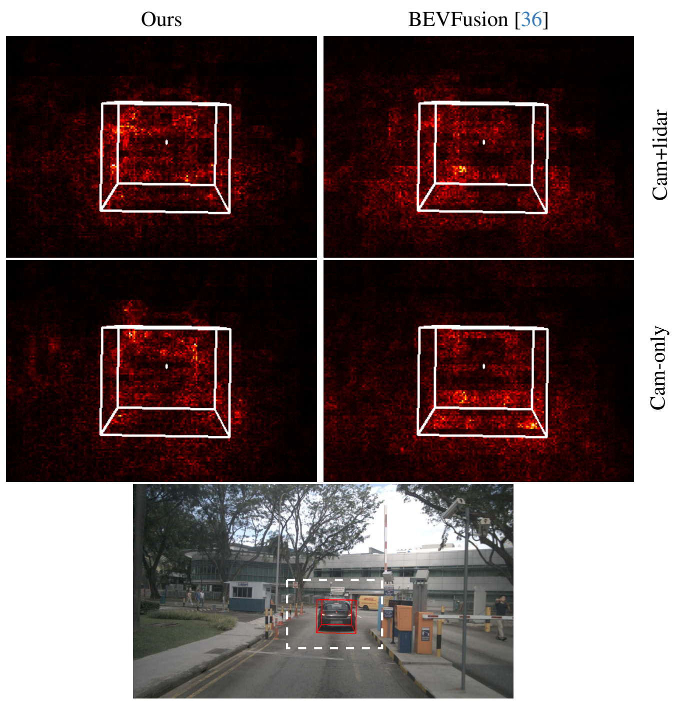

Camera
Lidar
End-to-end [36]
Supervised $\lambda = 0$
Supervised $\lambda = 0.001$
Supervised $\lambda = 0.01$
Supervised $\lambda = 0.1$
Supervised $\lambda = 1$
Supervised $\lambda = 5$
Supervised $\lambda = 100$

Figure S1. Depth maps obtained after different levels of depth supervision on an example from the nuScenes val set.

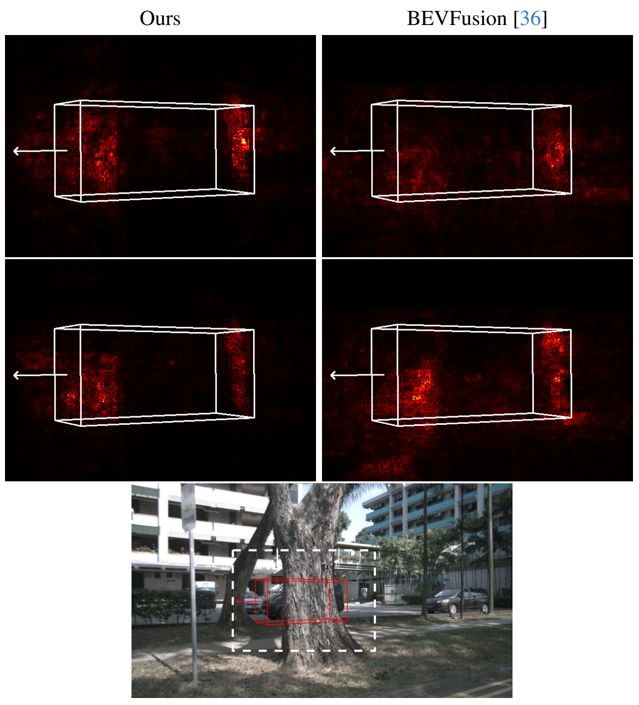
with WBF for each cell resolution, and then apply another WBF on the outputs from TTA of each model to get the final detections which we use for evaluations. For rotation augmentation, we use (-12.5, -6.25, 0, 6.25, 12.5) degrees.



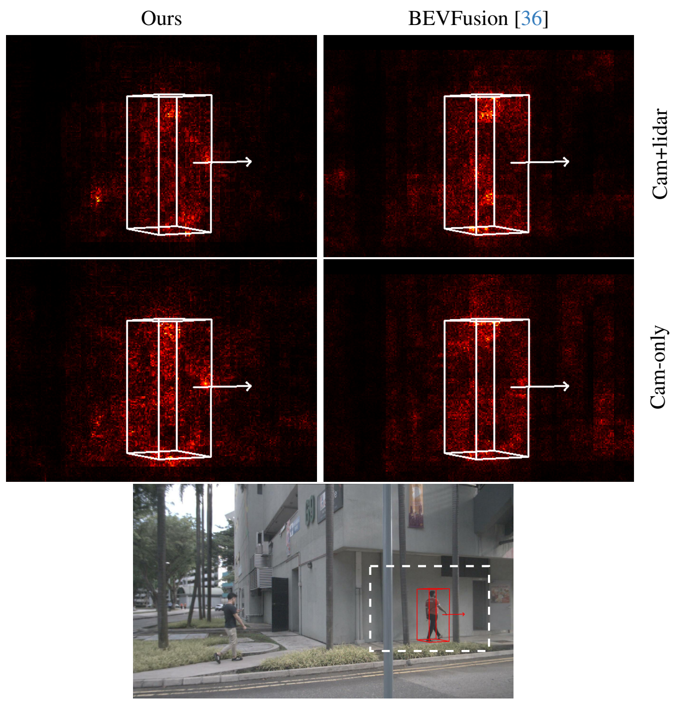
(a)



(b)



(c)



(d)

Figure S2. **(a)** Similar attention pattern as highlighted in the main text for a close object which is well-represented by the lidar point cloud. **(b)** For the same object, but at a later frame when the car moved further away from the ego, our model attends to the same area when trained with camera and lidar as when trained with camera only (left). **(c)** For an occluded object, whose representation in the lidar point cloud is weaker, our model attends to the entire unoccluded area in both settings. BEVFusion [36] appears to consistently attend to a larger neighbourhood of pixels. **(d)** The pedestrian, who is not well-represented by the lidar point cloud, is fully attended by our model in the presence of lidar and camera.

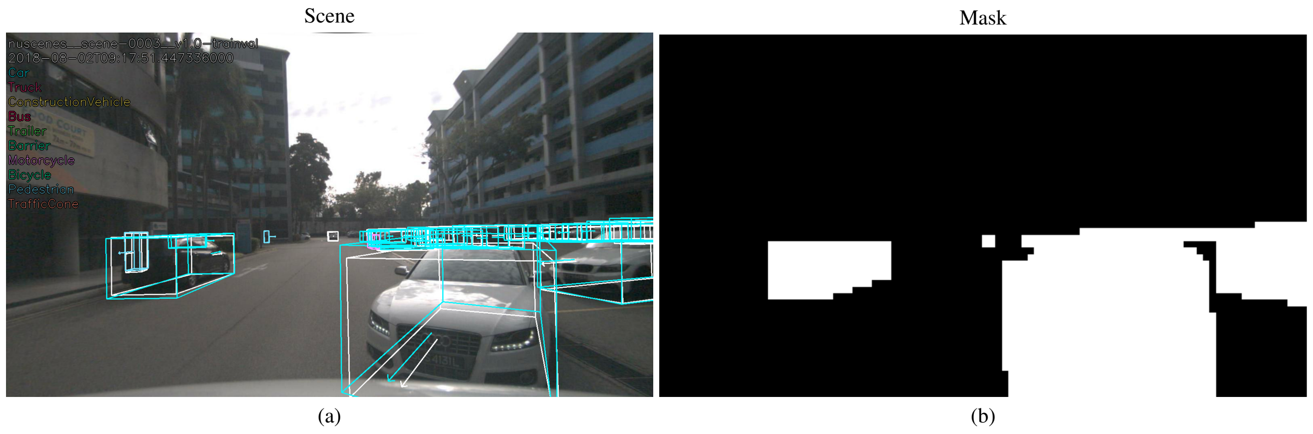


Figure S3. **(a)** Camera image I_i with annotations highlighted in white and our model's predictions, in colour. **(b)** Binary image Mask$^{(i)}$ created by in-painting the annotations.

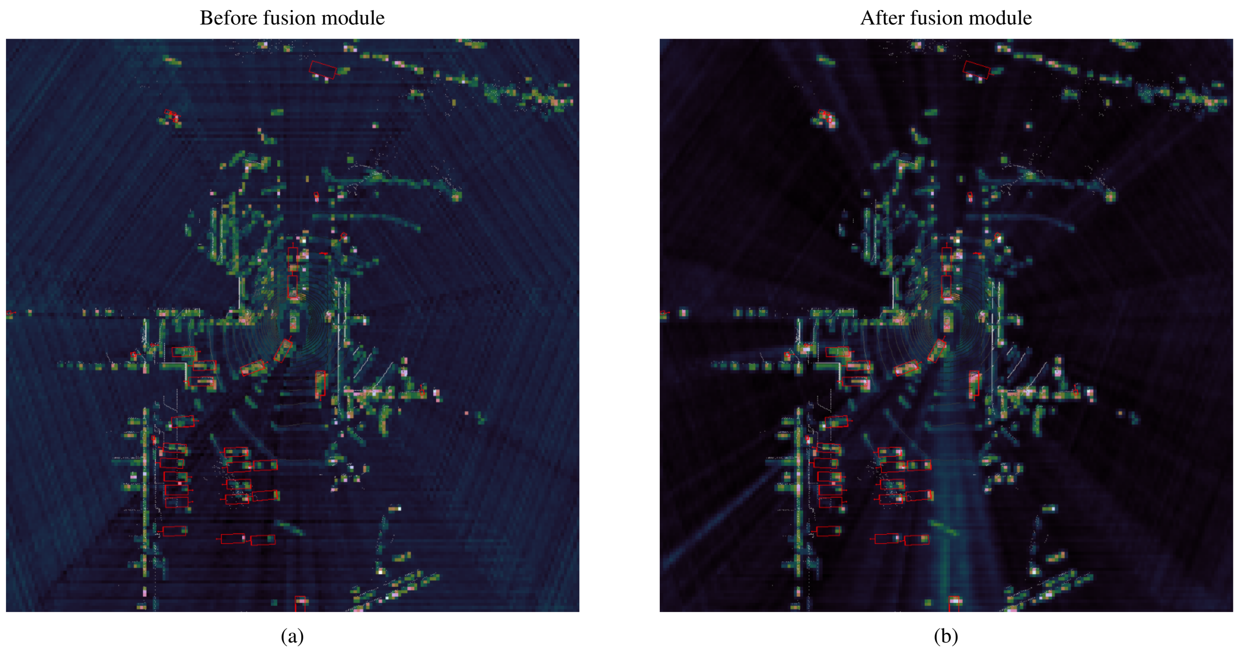
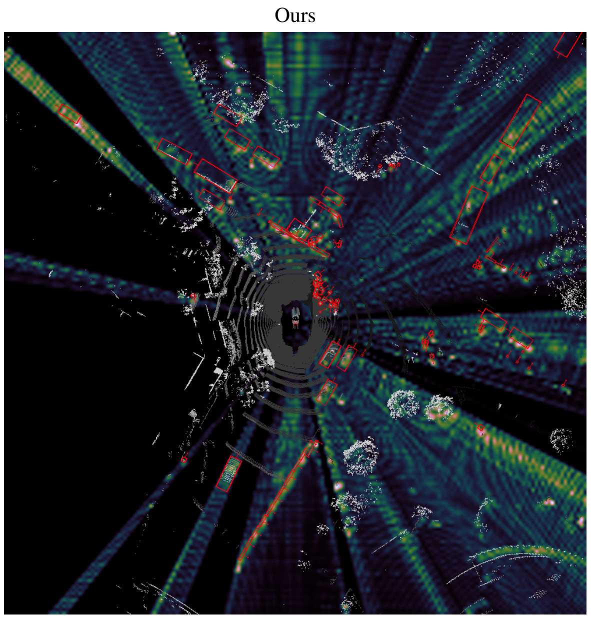
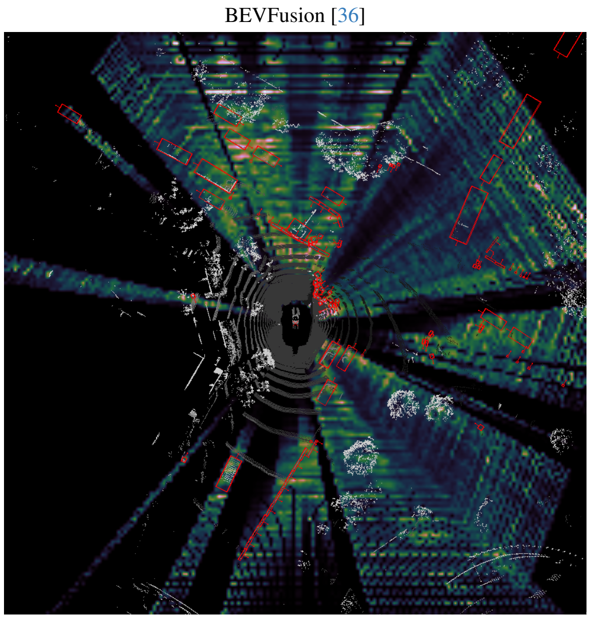


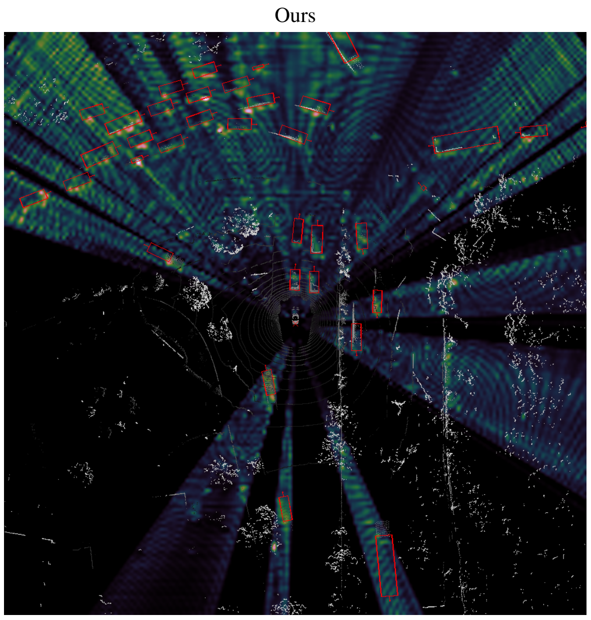
Figure S4. Activations in BEV space derived by summing up feature maps along the channel dimension. **(a)** uses the channel-wise concatenation of lidar and projected camera features. **(b)** uses the output of the fusion module, demonstrating its efficacy in suppressing background activations.



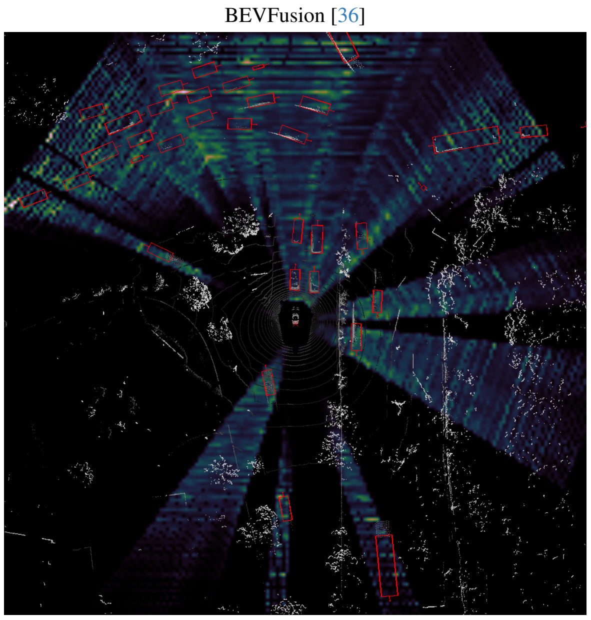
(a)



(b)



(c)



(d)

Figure S5. Additional examples showcasing the weight of projected camera features onto the BEV space. All examples presented in the analysis are from the validation set.