2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Multimodal Object Query Initialization for 3D Object Detection

Mathijs R. van Geerenstein[1,2,*], Felicia Ruppel[1,3,*], Klaus Dietmayer[3] and Dariu M. Gavrila[2]

***Abstract*— 3D object detection models that exploit both LiDAR and camera sensor features are top performers in large-scale autonomous driving benchmarks. A transformer is a popular network architecture used for this task, in which so-called object queries act as candidate objects. Initializing these object queries based on current sensor inputs is a common practice. For this, existing methods strongly rely on LiDAR data however, and do not fully exploit image features. Besides, they introduce significant latency. To overcome these limitations we propose EfficientQ3M, an efficient, modular, and multimodal solution for object query initialization for transformer-based 3D object detection models. The proposed initialization method is combined with a "modality-balanced" transformer decoder where the queries can access all sensor modalities throughout the decoder. In experiments, we outperform the state of the art in transformer-based LiDAR object detection on the competitive nuScenes benchmark and showcase the benefits of input-dependent multimodal query initialization, while being more efficient than the available alternatives for LiDAR-camera initialization. The proposed method can be applied with any combination of sensor modalities as input, demonstrating its modularity.**

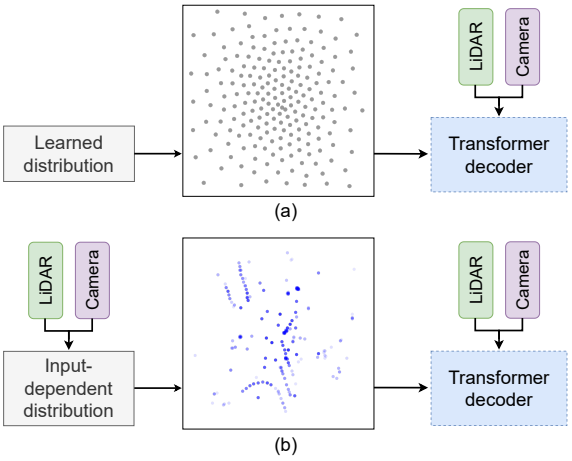

Fig. 1. Different query initialization approaches in transformer-based LiDAR-camera object detection. We show the initial object query locations from the bird's eye view with (a) input-agnostic initialization as in FUTR3D [3] (b) the proposed feature-informed initialization.

I. INTRODUCTION

3D object detection is a vital part of autonomous driving systems, and the resulting detections serve as a starting point for downstream tasks such as tracking or trajectory prediction. In automotive scenes, we try to predict multi-class 3D bounding boxes for all road users and other important objects around the ego vehicle. Models that exploit multi-modal sensor data are currently top performers in popular benchmarks for 3D object detection [1], [2]. The sensor suite typically consists of a roof-mounted LiDAR and a set of monocular cameras, the former providing a sparse 3D point cloud and the latter high-resolution dense images. These two sensor types are complementary: LiDAR brings accurate depth information and the cameras offer texture information and higher resolution for small, far-away objects.

In recent years, the transformer [4] architecture has successfully been applied to 2D [5]–[7] and 3D [3], [8]–[10] object detection tasks. Transformers rely on *object queries* to detect objects, where each query is an object candidate that can detect at most one object. A query is essentially a feature vector in a latent space that encodes all information needed to predict a classified bounding box. Usually, each query is accompanied by a reference or anchor point, with respect to which the bounding box is predicted [11]. The initialization of the query feature vectors and their locations is an active research topic, and we distinguish two approaches: learning a fixed, input-agnostic distribution for the object queries during training, or initializing them based on the current sensor inputs. Fig. 1 shows the learned distribution of initial query locations from input-agnostic method FUTR3D [3] (a), and input-dependent initialization using our proposed method (b).

Input-agnostic initialization generally requires many queries and multiple passes through the transformer decoder to achieve strong detection performance. Input-dependent initialization can improve on this by placing the queries at locations where we expect to find objects after predictions from a first-stage network. The concept of input-dependent query initialization in itself is not novel: it was proposed in [7] for the 2D domain and applied in TransFusion [10] for LiDAR-camera 3D detection. This state-of-the-art method has also been used as the query initialization approach for other models like DeepInteraction [12]. We however find two limitations with TransFusion's initialization that lead us to propose a new method. First, TransFusion takes a sequential approach to sensor fusion, where camera features are only used to refine a set of initial LiDAR-only predictions (Fig. 2a). This limits the benefit of the camera modality. Second, TransFusion uses an elaborate transformer network solely to fuse LiDAR and camera features into a shared bird's eye view (BEV) space from which the initial query locations are predicted. This adds significant overhead to the model.

To overcome these drawbacks, we propose a novel input-

*Equal contribution
[1]Robert Bosch GmbH, Corporate Research, 71272 Renningen, Germany, {firstname.lastname}@de.bosch.com
[2]Intelligent Vehicles Group, Delft University of Technology, the Netherlands, {initials.lastname}@{student.}tudelft.nl
[3]Institute of Measurement, Control and Microtechnology, Ulm University, Germany, {firstname.lastname}@uni-ulm.de

aware initialization approach to initialize object queries with both LiDAR and camera features while introducing only minimal computational overhead. Our method is combined with a *modality-balanced* transformer decoder (Fig. 2b), where the object queries can access both sensor modalities in each decoder layer.

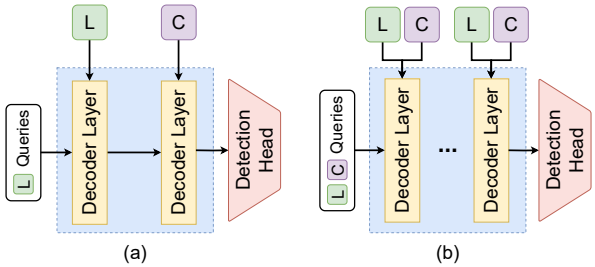

Fig. 2. Different approaches to sensor fusion within a transformer decoder. We call (a) sequential fusion, found in TransFusion [10] and (b) modality-balanced fusion in our proposed method. L is LiDAR and C is camera.

Our main contributions are as follows. We propose a novel input-dependent object query initialization strategy to initialize the query feature vectors with both LiDAR and camera features in a modality-balanced way, which is also able to work with any other combination of sensor modalities. We combine the proposed initialization with a modality-balanced transformer decoder to achieve state-of-the-art performance on the nuScenes [1] benchmark. The proposed initialization outperforms alternative solutions while being more efficient than the query initialization in state-of-the-art method TransFusion [10].

II. Related Work

Our work relates to two main disciplines in 3D object detection: LiDAR-only and LiDAR-camera detection. In addition, we elaborate on related methods for object query initialization with transformer-based object detection models.

A. LiDAR-based 3D Detection

For unimodal models, LiDAR-based detectors top the tables of popular benchmarks like nuScenes [1] and Waymo [2]. Most of these detectors quantize the LiDAR point cloud into a regular grid of 3D voxels [13], [14], 2D pillars [15] or perspective range views [16]–[18] using convolutional backbones. Others operate directly on the unordered, irregular point cloud [8], [19]–[22]. Many detectors adopt anchor boxes in their detection heads [13], [15], [23], where objects are predicted as offsets to these anchor boxes. Alternatively, center-based detectors [24], [25] capture objects as points and predict the 3D bounding box from this center point representation.

After the pioneering work of DETR [5] that applied transformers to 2D detection, transformer models have also found their way to 3D object detection. Contrary to DETR [5], where the model directly predicts the bounding box location in global image coordinates, most transformers for 3D detection predict boxes relative to an *anchor point*. These anchor locations are either fixed and independent of the current input [3], or computed using the current point cloud by a sampling method [8], [26] or center heatmap approach [10], [24], [25].

B. LiDAR-Camera 3D Detection

Early works in LiDAR-camera sensor fusion mainly adopt proposal-level fusion [27], [28], where proposals are generated in both modalities individually and then shared to the other(s) by projection. Following PointPainting [29], other works similarly apply semantic segmentation networks on images to augment point clouds with richer features [30]–[32]. These methods are better able to exploit the multimodal features but are more sensitive to feature alignment issues from suboptimal sensor calibration because of the hard association between points and pixels. Finally, there are methods that fuse both modalities into a shared BEV space, either with a direct BEV projection (view transform) of the image pixels [33]–[38], or by explicitly *lifting* image pixels into 3D space using projected LiDAR depth information [12], [39]–[42]. Within transformer-based models, we see two main approaches for multimodal fusion: those that deploy transformers only as the fusion mechanism for the sensor features [32], [36], [43], [44] and those that use transformers for both sensor fusion and the actual object detection [3], [10], [12], [45]. The proposed method is based on the latter principle.

C. Object Query Initialization

Many advances in query initialization originate from 2D object detectors in the image domain. In DETR's [5] initial implementation, the object queries are a small set of $M = 100$ learned embeddings with length $d = 256$, which form the object queries $\{\mathbf{q}_i\}_{i=1}^{M} \in \mathbb{R}^d$. Deformable DETR [6] adds positional information to the object queries so that predictions are made as offsets relative to the query positions, rather than globally. The object queries $\{\mathbf{q}_i\}_{i=1}^{M} \in \mathbb{R}^d$ are now accompanied by their 2D locations $\{\mathbf{c}_i\}_{i=1}^{M} \in \mathbb{R}^2$. In either case, the queries are learned and independent of the current sensor inputs at test time. Efficient DETR [7] applies input-dependent initialization with a region proposal network (RPN) to transformer-based 2D detection.

In 3D object detection, the total area spanned by the scene is very large relative to the size of objects. FUTR3D [3] operates with learned object queries and reference points similar to deformable DETR [6], but uses 3D locations and increases the number of object queries to $M = 900$ to get sufficient coverage of the large space. Other works [8], [26] use farthest point sampling [20] on the input point cloud to evenly spread query locations based on the current input. Finally, there are methods [10], [12], [25] most related to ours with input-informed query initialization, where the 2D BEV query locations $\{\mathbf{c}_i\}_{i=1}^{M} \in \mathbb{R}^2$ are taken as the top-M peaks in a predicted heatmap, and the feature vectors $\{\mathbf{q}_i\}_{i=1}^{M} \in \mathbb{R}^d$ are initialized with LiDAR features sampled at the locations. Different from them, the proposed method results in 3D initial locations $\{\mathbf{c}_i\}_{i=1}^{M} \in \mathbb{R}^3$, adds minimal overhead while incorporating modality-balanced fusion and initializes the query vectors with both LiDAR and camera features.

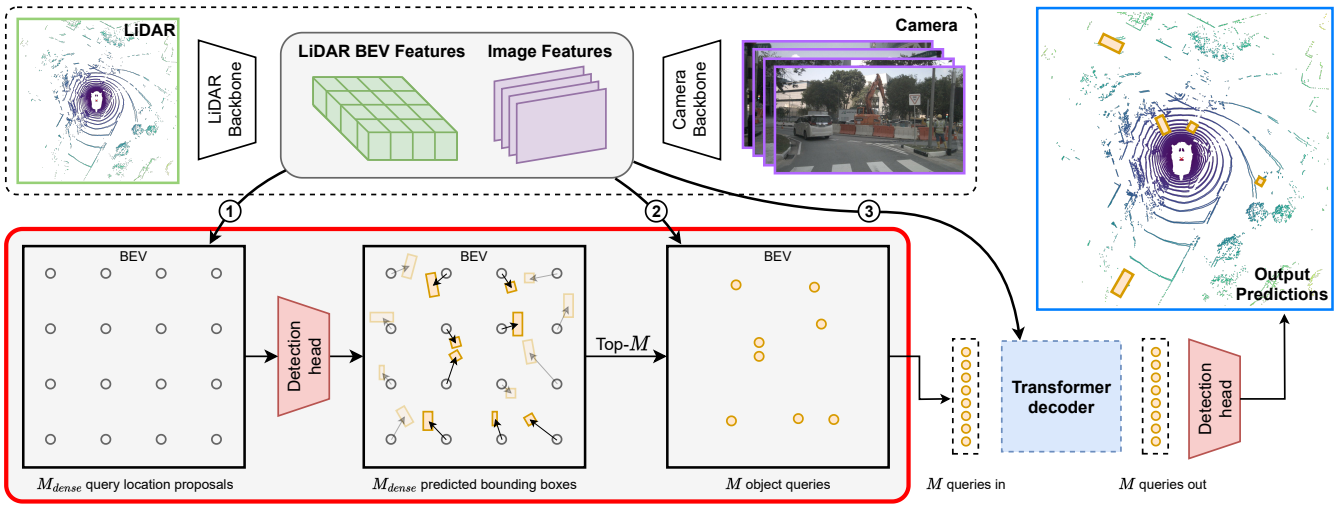

Fig. 3. Overview of EfficientQ3M, with the main contribution framed in **red**. We start with a fixed grid $\mathcal{C}$ of M_{dense} query location proposals. We sample LiDAR and camera features at instance level for each proposal ①, and predict a bounding box relative to the grid location. The 3D xyz centers of the top-M bounding boxes with the highest confidence scores are selected as the set of initial object query locations. We re-sample LiDAR and camera features for these M object queries ② and pass them to the modality-balanced decoder, where the queries have access to both sensor modalities in each layer of the decoder ③. A regression and classification head is used to produce the final detections from the object queries at the output of the decoder.

III. Methodology

In this section, we explain EfficientQ3M, a new multi-modal method for input-dependent object query initialization, where object queries are initialized with both LiDAR and camera features sampled at predicted 3D locations. An overview of our proposed model is presented in Fig. 3.

A. Multimodal Input-Dependent Initialization

In essence, we adopt a lightweight network to predict initial object locations from a large set of M_{dense} proposal object queries, and base the initial object queries for the transformer decoder on the top-M proposals. We choose M_{dense} to be much larger than M in order to have a sufficient number of proposals to cover the space densely and to not miss any objects. The initialization is lightweight, therefore it can be performed with a large number of M_{dense} queries, whereas the number of queries M in the decoder following the initialization is kept small. The proposed method is explained below in more detail.

We start by creating a dense grid of query proposal locations $\mathcal{C} \in \mathbb{R}^{X \times Y \times 1}$, spread out uniformly over the detection range in the (x, y) direction. Each 2D location in grid $\mathcal{C}$ is assigned the same fixed height to get 3D query locations. We now have a set of $M_{dense} = X \cdot Y$ initial query proposal locations, which are independent of the current sensor inputs. For all M_{dense} proposals, we sample sensor features at instance level from the given sensor feature maps after computing the respective view projections (Fig. 3 ①) using available intrinsic and extrinsic sensor parameters. The sensor features are fused, resulting in the corresponding query feature vectors. Queries are then made location-aware by adding a positional embedding based on their location using a sine encoding, following DETR [5]. The query feature vectors now contain the information needed to predict a 3D bounding box relative to their respective locations. We predict a classified bounding box from all M_{dense} query proposals, using the same regression head Φ_{reg} and classification head Φ_{cls} found later in the transformer decoder layers. From all proposal bounding boxes, we select the top-M with the highest confidence scores, and let the queries from which they are predicted be our set of initial object queries.

For each query $\mathbf{q}_i$ out of the top-M proposal queries, the location $\mathbf{c}_i$ is updated with the predicted 3D offset $\Delta\mathbf{x}_i$ from the regression head, i.e. $\mathbf{c}'_i = \mathbf{c}_i + \Delta\mathbf{x}_i$. We finally generate new query feature vectors from the updated locations, where the query feature vector $\mathbf{q}_i$ is composed by re-sampling features at the new location $\mathbf{c}'_i$ (Fig. 3 ②). This results in the object query feature vectors $\{\mathbf{q}_i\}_{i=1}^{M} \in \mathbb{R}^d$ and their 3D locations $\{\mathbf{c}'_i\}_{i=1}^{M} \in \mathbb{R}^3$, which we pass to the decoder. In the decoder, the object queries interact through self-attention, and have access to the sensor features from all modalities in cross-attention (Fig. 3 ③).

Because we predict a 3D offset $\Delta\mathbf{x}_i$ with Φ_{reg}, we deem it not necessary to initialize a 3D dense grid $\mathcal{C} \in \mathbb{R}^{X \times Y \times Z}$ with multiple different heights as proposals.

a) Modular and Multimodal: Our initialization method is modular because it can work with any sensor combination, such as camera-only, camera-RADAR, LiDAR-only and LiDAR-camera, similar to the modality-balanced decoder in FUTR3D [3]. The initial grid $\mathcal{C}$ with the proposal locations is identical in each case, but the modalities of features sampled at the locations will differ. Sensor features are sampled from the corresponding feature map around the projected query location. If there are multiple sensor modalities available, we concatenate the sampled features from both and fuse them as follows, exemplary for LiDAR-camera fusion:

$$\mathcal{SF}^i_{\text{fus}} = \Phi_{\text{fus}} \left(\mathcal{SF}^i_{\text{lid}} \oplus \mathcal{SF}^i_{\text{cam}} \right). \quad (1)$$

Here, $\mathcal{SF}^i_{\text{lid}}$ are the sampled LiDAR features, $\mathcal{SF}^i_{\text{cam}}$ the camera features, Φ_{fus} the fusion multi-layer perceptron (MLP) and $\mathcal{SF}^i_{\text{fus}}$ is the fused feature vector for query $\mathbf{q}_i$. When there is only one sensor modality available (e.g. LiDAR-only), Φ_{fus} is simply a linear projection.

In contrast to TransFusion [10], where an elaborate transformer network is used to create a shared LiDAR-camera heatmap to initialize the object queries, our implementation is lightweight and straightforward. On top of that, it is flexible and not specific to any sensor suite.

b) Model Details: The implementation of the backbones, transformer decoder, and final detection head – i.e. all components outside of the red frame in Fig. 3 – follows FUTR3D [3]. Our initialization method may be applied to other decoder designs with any combination of sensor modalities, as long as there exist transformations from global 3D coordinates to the respective sensor feature map coordinates.

B. Losses

We supervise the model in three locations, starting with the object query initialization. As explained in Sec. III-A, we predict a large set of M_{dense} bounding boxes and initialize our object queries from the top-M with the highest confidence score. To supervise the M_{dense} bounding boxes, we perform bipartite matching between the ground truth objects and all M_{dense} predicted boxes to get a set of one-to-one matches. The Hungarian algorithm [46] is used to produce these matches. The associated matching cost is a weighted sum of classification and regression costs:

$$C_{match} = \lambda_1 L_{cls}(\hat{p}, p) + \lambda_2 L_{reg}(\hat{b}, b) \tag{2}$$

where $(\hat{p}, \hat{b})$ are the predicted class confidence scores and bounding box parameters and (p, b) the corresponding supervisory signals, L_{cls} is the focal loss [47] and L_{reg} the L1 regression loss, and λ_1, λ_2 are the corresponding weights. For the set of matched predictions, we again compute the classification loss (focal loss) and regression loss (L1 loss).

Additionally, we supervise all M_{dense} predictions with a dense heatmap to improve convergence, because the number of matched predictions is much smaller than the number of proposals M_{dense}. For this, we take the class confidence scores from all predictions as a class-specific dense heatmap $\hat{\mathcal{S}} \in \mathbb{R}^{X \times Y \times K}$, which is supervised by a ground truth heatmap $\mathcal{S} \in \mathbb{R}^{X \times Y \times K}$ with the penalty reduced focal loss. Here, $X \times Y$ defines the spatial dimension of the heatmap, which matches our dense grid of query locations $\mathcal{C} \in \mathbb{R}^{X \times Y}$ and K is the number of classes. We take the K confidence scores for each prediction from grid $\mathcal{C}$ to obtain heatmap $\hat{\mathcal{S}}$. The ground truth heatmap is computed following CenterPoint [24]. We find that without this dense loss term, our method does not converge.

Next, the predicted bounding boxes at the output of each transformer decoder layer are supervised as in FUTR3D [3]. Finally, because sparse supervision can hinder learning in transformer-based models [48], we follow FUTR3D and implement an auxiliary detection head parallel to the transformer decoder for improved supervision of the LiDAR backbone. This CenterPoint [24] head is only used to help the LiDAR backbone learn better features during training, and is removed at test time.

IV. Experiments

In this section, we introduce the chosen dataset and implementation details before presenting the main results and ablation studies.

A. Dataset and Metrics

We use the large-scale autonomous driving dataset nuScenes [1] to evaluate our model. The dataset is a collection of 1000 *scenes*, each 20 seconds long and annotated at 2 Hz. Objects are annotated with 3D bounding boxes and a class label out of $K = 10$ object classes. Performance is measured with the popular mean average precision (mAP) metric and the custom nuScenes detection score (NDS).

B. Implementation Details

Our implementation is written using PyTorch [49] in the open-source MMDetection3D [50] framework. The code is built on top of the public release of FUTR3D [3], and we follow their model settings for the hyperparameters, unless stated otherwise. The most important details are listed below.

1) Model Settings: The LiDAR backbone is a VoxelNet [13], [14] with a voxel size of $(0.075\,\text{m}, 0.075\,\text{m}, 0.2\,\text{m})$. The image backbone is a VoVNET [51] and is pre-trained on nuScenes [1] using the camera-only version of FUTR3D. The number of object query proposals is $M_{dense} = 3600$, spread out uniformly over the (x, y) detection range. We report results for both $M = 200$ and 900 object queries.

2) Training Strategy: We adopt a common augmentation pipeline for the LiDAR data, where we use random rotation with $r \in [\pi/4, \pi/4]$, random scaling with $s \in [0.9, 1.1]$, random xyz translation with a standard deviation of 0.5, and random horizontal and vertical flipping. We use CBGS [52] class-balanced sampling to improve the class balance in nuScenes. Finally, we use ground truth copy-paste augmentation [14], and disable it for the final epochs to match the real data distribution again [30]. We do not adopt any augmentation at test time.

We first train the LiDAR branch of our model, using a pre-trained LiDAR backbone. The schedule is set to 6 epochs, with GT copy-paste augmentation disabled in the final 3 epochs. From there, the pre-trained image backbone is added and the LiDAR-camera model is trained for another 6 epochs with both backbones frozen. Such a sequential approach has shown to yield better performance than joint training from the start, because it allows for better augmentation in the LiDAR-only stage of training [3], [10]. We use an initial learning rate of 1.0×10^{-4} for both LiDAR-only and LiDAR-camera training, with a cyclic learning rate policy [53].

C. Main Results

The detection performance of EfficientQ3M is evaluated on both the nuScenes *validation* and *test* set and the results are presented in Tab. I. On the nuScenes *val* set,

TABLE I

COMPARISON TO THE STATE OF THE ART ON THE NUSCENES *validation* AND *test* SET. THE BEST SCORES FOR EACH SENSOR MODALITY ARE **BOLD**.

Method	Modality	Backbone		*validation*		*test*	
		Camera	LiDAR	mAP ↑	NDS ↑	mAP ↑	NDS ↑
CenterPoint [24]	L	-	VoxelNet	59.6	66.8	60.3	67.3
TransFusion-L [10]	L	-	VoxelNet	65.1	**69.9**	65.5	70.2
FUTR3D [3]	L	-	VoxelNet	63.7	69.0	65.3	69.9
EfficientQ3M (ours)	L	-	VoxelNet	**65.3**	69.6	**66.1**	**70.3**
PointAugmenting [30]	L+C	DLA34	VoxelNet	-	-	66.8	71.0
TransFusion [10]	L+C	DLA34	VoxelNet	67.3	70.9	68.9	71.6
DeepInteraction [12]	L+C	R50	VoxelNet	69.9	72.6	**70.8**	**73.4**
FUTR3D [3]	L+C	VoVNet	VoxelNet	70.3	73.1	69.4	72.1
EfficientQ3M (ours)	L+C	VoVNet	VoxelNet	**71.2**	**73.5**	70.5	72.6

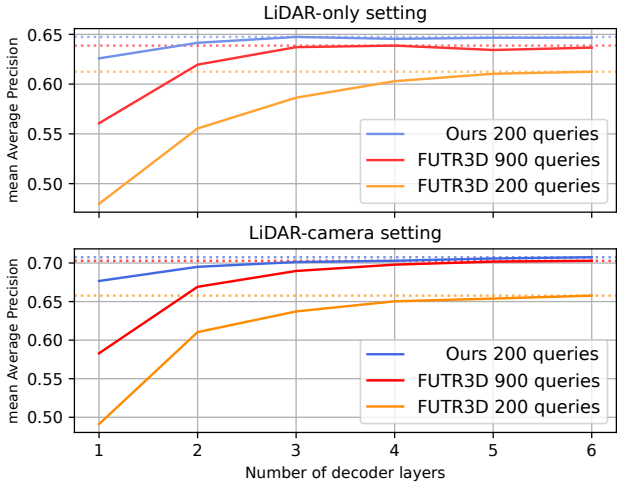

Fig. 4. Detection performance vs. the number of decoder layers on the nuScenes *val* set in the LiDAR-only and LiDAR-camera setting, compared to FUTR3D [3]. The proposed method needs fewer layers and fewer queries to outperform FUTR3D.

the proposed method outperforms the state of the art in transformer-based object detection for both the LiDAR-only, as well as the LiDAR-camera fusion setting. This also applies for the *test* set with LiDAR-only detection. With LiDAR-camera fusion on the *test* set, EfficientQ3M outperforms its baseline FUTR3D [3] as well as TransFusion [10]. Note that, since DeepInteraction [12] utilizes TransFusion's initialization strategy, it may further benefit by incorporating the proposed modality-balanced method for intialization.

In Fig. 4, we compare our model to FUTR3D with a varying number of decoder layers. We find that the proposed input-dependent object query initialization not only produces superior detection scores, but also enables the use of fewer object queries and decoder layers. Where FUTR3D needs multiple passes through the decoder to achieve high performance, our input-dependent initialization already shows good performance for a single-layer model. Specifically, the proposed method with 200 object queries outperforms the baseline with 900 queries for any number of decoder layers. We also find that the input-agnostic baseline degrades strongly when decreasing the number of object queries.

TABLE II

COMPARISON OF DIFFERENT QUERY INITIALIZATION METHODS PAIRED WITH THE MODALITY-BALANCED DECODER ON THE NUSCENES *val* SET.

Init. Method	Mod.	#Q	mAP ↑	Lat. (ms) ↓	#P
Input-agnostic	L	900	63.7	208.6	7.29
w/ TransFusion	L	200	64.5 (+0.8)	212.7 (+2.0%)	7.89
w/ ours	L	200	64.7 (+1.0)	209.5 (+0.4%)	7.51
Input-agnostic	L+C	900	70.3	645.4	9.32
w/ TransFusion	L+C	200	70.3 (+0.0)	751.1 (+16.4%)	11.8
w/ ours	L+C	200	70.8 (+0.5)	652.3 (+1.1%)	9.80

In Tab. II we compare the proposed method to the initialization of the closest related work: TransFusion [10], with $M = 200$ queries. We include FUTR3D's [3] input-agnostic approach for reference, with $M = 900$ queries for a fair comparison. Only the object query initialization method is varied, all other model settings, such as the backbones, transformer decoder and detection head are equal. We find that both initialization strategies achieve similar improvements on the baseline in the LiDAR-only setting, but that the proposed method is superior for LiDAR-camera fusion. We contribute this to the benefit of our modality-balanced design and to predicting initial 3D query locations, which allow for more accurate sampling of image features, compared to the 2D approach in TransFusion.

Additionally, we find that the proposed method is lightweight and efficient compared to TransFusion's initialization. Especially for LiDAR-camera fusion, our method remains efficient with only $+6.9$ ms of added latency while TransFusion's introduces $+105.7$ ms of overhead compared to a model with input-agnostic queries (i.e. a $15\times$ difference in the initialization stage, highlighted in gray). This is explained by the elaborate transformer network used in their query initialization, the size of which also shows when looking at the number of model parameters #P (measured in millions, excluding the backbones).

Fig. 5 shows a sample of detections made with the proposed method. We highlight three successful detections in a difficult setting, i.e. with rain and strong occlusions.

Fig. 5. Example of output predictions with the proposed model on the nuScenes *val* set. The LiDAR BEV shows ground truth objects in dark green.

TABLE III

ABLATION ON OBJECT QUERY INITIALIZATION COMPONENTS ON THE NUSCENES *val* SET, IN THE LIDAR-ONLY SETTING WITH 200 QUERIES.

Query Initialization Strategy	#L.	mAP ↑	NDS ↑
Input-agnostic	6	60.9 ± 0.2	67.1 ± 0.1
Input-dependent Refs.	6	64.3 ± 0.1	**69.1** ± 0.1
Input-dependent Refs. + Feats.	6	**64.3** ± 0.1	69.0 ± 0.0
Input-agnostic	1	48.2 ± 0.2	56.8 ± 0.2
Input-dependent Refs.	1	61.8 ± 0.2	67.0 ± 0.2
Input-dependent Refs. + Feats.	1	**62.2** ± 0.1	**67.3** ± 0.1

TABLE IV

COMPARING DETECTION PERFORMANCE (MAP ↑) ON THE NUSCENES *val* SET FOR BOTH SENSOR SETUPS WITH 200 AND 900 OBJECT QUERIES.

	Mod.	# 200	# 900
FUTR3D	L	61.3 (-2.4)	63.7
FUTR3D	L+C	65.8 (-4.5)	70.3
EfficientQ3M (ours)	L	64.7 (-0.6)	65.3
EfficientQ3M (ours)	L+C	70.8 (-0.4)	71.2

D. Ablation

1) Query Initialization: We test if it suffices to initialize the query feature vectors with positional embeddings based on their predicted location (Refs.), or if we instead need to initialize them with sensor features sampled at their location (Refs. + Feats.), as proposed. We compare both approaches with the input-agnostic baseline for $M = 200$ object queries, with all methods trained on a reduced schedule, see Tab. III.

We find that, for 6 decoder layers, both versions of input-dependent initialization perform similarly: initializing the query feature vectors with sensor features does not bring obvious benefits. When we decrease the number of decoder layers to 1, however, the benefit of initializing the query vectors with sensor features is visible: it results in a small performance increase compared to using the positional embedding. Additionally, we see the improvement on the input-agnostic baseline increase. The baseline performance drops heavily because it needs multiple layers to iteratively update the query location if it is not located close to an object by chance at initialization.

2) Number of Queries: FUTR3D [3] uses $M = 900$ to get sufficient coverage of the large 3D space. With the proposed input-dependent query initialization, we are able to use fewer queries and still have them located close to the objects in the scene. We compare our method with FUTR3D for 200 and 900 queries, where we have fine-tuned FUTR3D's pretrained model to learn a new distribution with $M = 200$.

In Tab. IV, we find that the proposed method can still achieve strong performance even with many fewer queries, thanks to the input-dependent initialization. Specifically, our method with 200 queries outperforms FUTR3D with 900 queries. We do still see a marginal benefit of using 900 queries with the proposed initialization method, even though 200 queries are enough to cover the maximum number of objects in nuScenes. We hypothesize that this is caused by the larger self-attention operation in the decoder which allows for querying more information, and by the additional queries compensating for imperfect initialization.

V. CONCLUSION

We introduced EfficientQ3M, a novel and efficient approach for initializing object queries in transformer-based 3D object detection models from any sensor modality. Existing methods strongly rely on a LiDAR-only first stage, which limits the benefit from the camera domain. EfficientQ3M overcomes this limitation by initializing object queries with features from any combination of sensor modalities. Compared to an input-agnostic method, it achieves better performance with fewer queries and decoder layers. The proposed method, when combined with a modality-balanced transformer decoder, outperforms the state of the art for query-based LiDAR object detection in the nuScenes benchmark. Additionally, EfficientQ3M demonstrates significant efficiency gains compared to related work for query initialization. Furthermore, the modularity of the proposed method allows its application to different transformer decoders as well. For future work, it would be interesting to extend the scope of the experiments to obtain results for more sensor setups, such as RADAR-camera and camera-only.

REFERENCES

[1] 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," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, USA: IEEE, June 2020, pp. 11 618–11 628.

[2] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, "Scalability in Perception for Autonomous Driving: Waymo Open Dataset," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, USA: IEEE, June 2020, pp. 2443–2451.

[3] X. Chen, T. Zhang, Y. Wang, Y. Wang, and H. Zhao, "FUTR3D: A Unified Sensor Fusion Framework for 3D Detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2023, pp. 172–181.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017.

[5] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-End Object Detection with Transformers," in *Computer Vision – ECCV 2020*. Springer International Publishing, 2020, vol. 12346, pp. 213–229.

[6] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable DETR: Deformable Transformers for End-to-End Object Detection," in *International Conference on Learning Representations (ICLR)*, 2021.

[7] Z. Yao, J. Ai, B. Li, and C. Zhang, "Efficient DETR: Improving End-to-End Object Detector with Dense Prior," Apr. 2021, issue: arXiv:2104.01318 arXiv:2104.01318 [cs].

[8] I. Misra, R. Girdhar, and A. Joulin, "An End-to-End Transformer Model for 3D Object Detection," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. Montreal, QC, Canada: IEEE, Oct. 2021, pp. 2886–2897.

[9] F. Ruppel, F. Faion, C. Gläser, and K. Dietmayer, "Transformers for Object Detection in Large Point Clouds," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, Oct. 2022, pp. 832–838.

[10] 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 *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. New Orleans, LA, USA: IEEE, June 2022, pp. 1080–1089.

[11] J. Mao, S. Shi, X. Wang, and H. Li, "3D Object Detection for Autonomous Driving: A Comprehensive Survey," *International Journal of Computer Vision*, vol. 131, no. 8, pp. 1909–1963, Aug. 2023.

[12] Z. Yang, J. Chen, Z. Miao, W. Li, X. Zhu, and L. Zhang, "DeepInteraction: 3D Object Detection via Modality Interaction," *Advances in Neural Information Processing Systems*, vol. 35, pp. 1992–2005, Dec. 2022.

[13] Y. Zhou and O. Tuzel, "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4490–4499.

[14] Y. Yan, Y. Mao, and B. Li, "SECOND: Sparsely Embedded Convolutional Detection," *Sensors*, vol. 18, no. 10, p. 3337, Oct. 2018.

[15] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast Encoders for Object Detection From Point Clouds," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA: IEEE, June 2019, pp. 12 689–12 697.

[16] L. Fan, X. Xiong, F. Wang, N. Wang, and Z. Zhang, "RangeDet: In Defense of Range View for LiDAR-based 3D Object Detection," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. Montreal, QC, Canada: IEEE, Oct. 2021, pp. 2898–2907.

[17] Y. Chai, P. Sun, J. Ngiam, W. Wang, B. Caine, V. Vasudevan, X. Zhang, and D. Anguelov, "To the Point: Efficient 3D Object Detection in the Range Image with Graph Convolution Kernels," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Nashville, TN, USA: IEEE, June 2021, pp. 15 995–16 004.

[18] P. Sun, W. Wang, Y. Chai, G. Elsayed, A. Bewley, X. Zhang, C. Sminchisescu, and D. Anguelov, "RSN: Range Sparse Net for Efficient, Accurate LiDAR 3D Object Detection," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Nashville, TN, USA: IEEE, June 2021, pp. 5721–5730.

[19] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI: IEEE, July 2017, pp. 77–85.

[20] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017.

[21] Z. Yang, Y. Sun, S. Liu, and J. Jia, "3DSSD: Point-Based 3D Single Stage Object Detector," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, USA: IEEE, June 2020, pp. 11 037–11 045.

[22] Z. Li, F. Wang, and N. Wang, "LiDAR R-CNN: An Efficient and Universal 3D Object Detector," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Nashville, TN, USA: IEEE, June 2021, pp. 7542–7551.

[23] J. Deng, S. Shi, P. Li, W. Zhou, Y. Zhang, and H. Li, "Voxel R-CNN: Towards High Performance Voxel-based 3D Object Detection," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 2, pp. 1201–1209, May 2021, number: 2.

[24] T. Yin, X. Zhou, and P. Krahenbuhl, "Center-based 3D Object Detection and Tracking," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Nashville, TN, USA: IEEE, June 2021, pp. 11 779–11 788.

[25] Z. Zhou, X. Zhao, Y. Wang, P. Wang, and H. Foroosh, "CenterFormer: Center-Based Transformer for 3D Object Detection," *Computer Vision – ECCV 2022*, vol. 13698, pp. 496–513, 2022.

[26] F. Ruppel, F. Faion, C. Gläser, and K. Dietmayer, "Transformers for Multi-Object Tracking on Point Clouds," in *2022 IEEE Intelligent Vehicles Symposium (IV)*, June 2022, pp. 852–859.

[27] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3D Object Detection Network for Autonomous Driving," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI: IEEE, July 2017, pp. 6526–6534.

[28] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, "Joint 3D Proposal Generation and Object Detection from View Aggregation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 1–8, iSSN: 2153-0866.

[29] S. Vora, A. H. Lang, B. Helou, and O. Beijbom, "PointPainting: Sequential Fusion for 3D Object Detection," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, USA: IEEE, June 2020, pp. 4603–4611.

[30] C. Wang, C. Ma, M. Zhu, and X. Yang, "PointAugmenting: Cross-Modal Augmentation for 3D Object Detection," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Nashville, TN, USA: IEEE, June 2021, pp. 11 789–11 798.

[31] T. Huang, Z. Liu, X. Chen, and X. Bai, "EPNet: Enhancing Point Features with Image Semantics for 3D Object Detection," in *Computer Vision – ECCV 2020*. Cham: Springer International Publishing, 2020, vol. 12360, pp. 35–52.

[32] S. Xu, D. Zhou, J. Fang, J. Yin, Z. Bin, and L. Zhang, "FusionPainting: Multimodal Fusion with Adaptive Attention for 3D Object Detection," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, Sept. 2021, pp. 3047–3054.

[33] M. Liang, B. Yang, S. Wang, and R. Urtasun, "Deep Continuous Fusion for Multi-sensor 3D Object Detection," in *Computer Vision – ECCV 2018*. Cham: Springer International Publishing, 2018, vol. 11220, pp. 663–678.

[34] Z. Liu, H. Tang, A. Amini, X. Yang, H. Mao, D. L. Rus, and S. Han, "BEVFusion: Multi-Task Multi-Sensor Fusion with Unified Bird's-Eye View Representation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 2774–2781.

[35] F. Drews, D. Feng, F. Faion, L. Rosenbaum, M. Ulrich, and C. Gläser, "DeepFusion: A Robust and Modular 3D Object Detector for Lidars, Cameras and Radars," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 560–567.

[36] Y. Zeng, D. Zhang, C. Wang, Z. Miao, T. Liu, X. Zhan, D. Hao, and C. Ma, "LIFT: Learning 4D LiDAR Image Fusion Transformer for 3D Object Detection," in *2022 IEEE/CVF Conference on Computer*

Vision and Pattern Recognition (CVPR). New Orleans, LA, USA: IEEE, June 2022, pp. 17 151–17 160.

[37] Y. Li, Y. Chen, X. Qi, Z. Li, J. Sun, and J. Jia, "Unifying Voxel-based Representation with Transformer for 3D Object Detection," Oct. 2022, issue: arXiv:2206.00630 arXiv:2206.00630 [cs].

[38] J. H. Yoo, Y. Kim, J. Kim, and J. W. Choi, "3D-CVF: Generating Joint Camera and LiDAR Features Using Cross-view Spatial Feature Fusion for 3D Object Detection," in *Computer Vision – ECCV 2020*. Cham: Springer International Publishing, 2020, vol. 12372, pp. 720–736.

[39] 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," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 10 421–10 434.

[40] P. Jacobson, Y. Zhou, W. Zhan, M. Tomizuka, and M. C. Wu, "Center Feature Fusion: Selective Multi-Sensor Fusion of Center-based Objects," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 8312–8318.

[41] T. Yin, X. Zhou, and P. Krähenbühl, "Multimodal Virtual Point 3D Detection," in *Advances in Neural Information Processing Systems*, vol. 34. Curran Associates, Inc., 2021, pp. 16 494–16 507.

[42] Y. Jiao, Z. Jie, S. Chen, J. Chen, L. Ma, and Y.-G. Jiang, "MSMDFusion: A Gated Multi-Scale LiDAR-Camera Fusion Framework with Multi-Depth Seeds for 3D Object Detection," Nov. 2022, issue: arXiv:2209.03102 arXiv:2209.03102 [cs].

[43] Y. Yang, J. Liu, T. Huang, Q.-L. Han, G. Ma, and B. Zhu, "RaLiBEV: Radar and LiDAR BEV Fusion Learning for Anchor Box Free Object Detection System," Nov. 2022, issue: arXiv:2211.06108 arXiv:2211.06108 [cs].

[44] Y. Kim, K. Park, M. Kim, D. Kum, and J. W. Choi, "3D Dual-Fusion: Dual-Domain Dual-Query Camera-LiDAR Fusion for 3D Object Detection," Nov. 2022, issue: arXiv:2211.13529 arXiv:2211.13529 [cs].

[45] X. Xu, S. Dong, L. Ding, J. Wang, T. Xu, and J. Li, "FusionRCNN: LiDAR-Camera Fusion for Two-stage 3D Object Detection," Sept. 2022, issue: arXiv:2209.10733 arXiv:2209.10733 [cs].

[46] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[47] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal Loss for Dense Object Detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2980–2988.

[48] Z. Zong, G. Song, and Y. Liu, "DETRs with Collaborative Hybrid Assignments Training," Mar. 2023, issue: arXiv:2211.12860 arXiv:2211.12860 [cs].

[49] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," *31st Conference on Neural Information Processing Systems (NIPS)*, no. NeurIPS-W, 2017, number: NeurIPS-W.

[50] MMDetection3D Contributors, "OpenMMLab's Next-generation Platform for General 3D Object Detection," July 2020.

[51] Y. Lee, J.-w. Hwang, S. Lee, Y. Bae, and J. Park, "An Energy and GPU-Computation Efficient Backbone Network for Real-Time Object Detection," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Long Beach, CA, USA: IEEE, June 2019, pp. 752–760.

[52] B. Zhu, Z. Jiang, X. Zhou, Z. Li, and G. Yu, "Class-balanced Grouping and Sampling for Point Cloud 3D Object Detection," Aug. 2019, issue: arXiv:1908.09492 arXiv:1908.09492 [cs].

[53] L. N. Smith, "Cyclical Learning Rates for Training Neural Networks," Apr. 2017, issue: arXiv:1506.01186 arXiv:1506.01186 [cs].