Cross Modal Transformer: Towards Fast and Robust 3D Object Detection

Junjie Yan Yingfei Liu Jianjian Sun Fan Jia Shuailin Li
Tiancai Wang Xiangyu Zhang
MEGVII Technology

Abstract

In this paper, we propose a robust 3D detector, named Cross Modal Transformer (CMT), for end-to-end 3D multi-modal detection. Without explicit view transformation, CMT takes the image and point clouds tokens as inputs and directly outputs accurate 3D bounding boxes. The spatial alignment of multi-modal tokens is performed by encoding the 3D points into multi-modal features. The core design of CMT is quite simple while its performance is impressive. It achieves 74.1% NDS (state-of-the-art with single model) on nuScenes test set while maintaining faster inference speed. Moreover, CMT has a strong robustness even if the LiDAR is missing. Code is released at https://github.com/junjie18/CMT.

1. Introduction

Multi-sensor fusion has shown its great superiority in autonomous driving system [31, 8, 22, 1, 27]. Different sensors usually provide the complementary information for each other. For instance, the camera captures information in a perspective view and the image contains rich semantic features while point clouds provide much more localization and geometry information. Taking full advantage of different sensors helps reduce the uncertainty and makes accurate and robust prediction.

Sensor data of different modalities usually has large discrepancy in distribution, making it hard to merge the multi-modalities. State-of-the-art (SoTA) methods tend to fuse the multi-modality by constructing unified bird’s-eye-view (BEV) representation [31, 27, 22] or querying from tokens [1, 8]. For example, BEVFusion [31] explores a unified representation by BEV transformation for BEV feature fusion (see Fig. 1(a)). TransFusion [1] follows a two-stage pipeline and the camera images in second stage provide supplementary information for prediction refinement (see Fig. 1(b)). However, exploring a truly end-to-end pipeline for multi-sensor fusion remains to be a question.

Figure 1: Comparison between BEVFusion, TransFusion, and our proposed CMT. (a) In BEVFusion, the camera features are transformed into BEV space by view transform. Two modality features are concatenated in BEV space and the BEV encoder is adopted for fusion. (b) TransFusion first generates the queries from the high response regions of LiDAR features. After that, object queries interact with point cloud features and image features separately. (c) In CMT, the object queries directly interact with multi modality features simultaneously. Position encoding (PE) is added to the multi-modal features for alignment. “VT” is the view transformation from image to 3D space.

Recently, the effectiveness of end-to-end object detection with transformer (DETR) [3, 60] has been proved in many perception tasks, such as instance segmentation [13, 15], multi-object tracking [55, 33] and visual 3D detection [47, 29, 30]. The DETR architecture is simple yet effective thanks to the object queries for representing different

✉ Corresponding author.
instances and bipartite matching for one-to-one assignment.

Inspired by DETR, we aim to build an elegant end-to-end pipeline for multi-modal fusion in 3D object detection. In DETR, object queries directly interact with the image tokens through cross-attention in transformer decoder. For 3D object detection, one intuitive way is to concatenate the image and point cloud tokens together for further interaction with object queries. However, the concatenated tokens are disordered and unaware of their corresponding locations in 3D space. Therefore, it is necessary to provide the location prior for multi-modal tokens and object queries.

In this paper, we propose Cross-Modal Transformer (CMT), a simple yet effective end-to-end pipeline for robust 3D object detection (see Fig. 1(c)). First, we propose the Coordinates Encoding Module (CEM), which produces position-aware features, by encoding 3D points set implicitly into multi-modal tokens. Specifically, for camera images, 3D points sampled from frustum space are used to indicate the probability of 3D positions for each pixel. While for LiDAR, the BEV coordinates are simply encoded into the point cloud tokens. Next, we introduce the position-guided queries. Each query is initialized as a 3D reference point following PETR [29]. We transform the 3D coordinates of reference points to both image and LiDAR spaces, to perform the relative coordinates encoding in each space.

The proposed CMT framework brings many advantages compared to existing methods. Firstly, our method is a simple and end-to-end pipeline and can be easily extended. The 3D positions are encoded into the multi-modal features implicitly, which avoids introducing the bias caused by explicit cross-view feature alignment. Secondly, our method only contains basic operations, without the feature sampling or complex 2D-to-3D view transformation on multi-modal features. It achieves state-of-the-art performance and shows obvious superiority compared to existing approaches, as shown in the left graph of Fig. 2. Thirdly, the robustness of our CMT is much stronger than other existing approaches. Extremely, under the condition of LiDAR miss, our CMT with only image tokens can achieve similar performance compared to those vision-based 3D object detectors [29, 26] (see the right graph of Fig. 2).

To summarize, our contributions are:

- we propose a fast and robust 3D detector, which is a truly end-to-end framework without any post-process. It overcomes the sensor missing problem.
- The 3D positions are encoded into the multi-modal tokens, without any complex operations, like grid sampling and voxel-pooling.
- CMT achieves state-of-the-art 3D detection performance on nuScenes dataset. It provides a simple baseline for future research.

2. Related Work

2.1. Camera Based 3D Object Detection

Camera-based 3D object detection is one of the basic tasks in computer vision. Early works [45, 44] mainly follow the dense prediction pipeline. They first localize the objects on image plane and then predict their relevant 3D attributes, such as depth, size and orientation. However, with the surrounding cameras, the perspective-view based design requires elaborate post-processes to eliminate the redundant predictions of the overlapping regions. Recently, 3D object detection under the BEV has attracted increasing attention. The BEV representation provides a unified coordinate to fuse information from multiple camera views. LSS [35], BEVDet [17] and BEVDepth [24] predict the depth distribution to lift the image features to 3D frustum meshgrid. Besides, inspired by DETR [4], DETR3D [47] and BEVFormer [26] project the predefined BEV queries onto images and then employ the transformer attention to model
the relation of multi-view features. The above methods explicitly project the local image feature from 2D perspective view to BEV. Different from them, PETR [29, 30] and SpatialDETR [12] adopt the positional embedding that depends on the camera poses, allowing the transformer to implicitly learn the projection from image views to 3D space.

2.2. LiDAR Based 3D Object Detection

LiDAR-based 3D object detection aims to predict 3D object bounding boxes using the point clouds captured from LiDAR. Existing methods process the point cloud into different representations. Point-based methods [36, 37, 38, 39, 25, 53] directly extract features from raw point clouds and predict 3D bounding boxes. PointNet [37] is the first architecture to process the point cloud in an end-to-end manner, which preserves the spatial characteristics of the point cloud. Other methods project the unordered, irregular LiDAR point clouds onto a regular feature space such as 3D voxels [58, 51, 9, 10], feature pillars [19, 46, 54] and range images [14, 41]. Then the features are extracted in the BEV plane using a standard 2D backbone. VoxelNet [58] first divides the raw point clouds into regular voxel grids, and then uses PointNet network to extract features from the points in each voxel grid.

2.3. Multi-modal 3D Object Detection

Multi-sensor fusion in 3D detection has gained great attention in recent years. State-of-the-art (SoTA) methods tend to find a unified representation for both modalities, or define object queries to fuse the features for further prediction. For example, BEVFusion [31, 27] applies a lift-splat-shoot (LSS) operation to project image feature onto BEV space and concatenates it with LiDAR feature. UVTR [22] generates a unified representation in the 3D voxel space by deformable attention [60]. While for query-based methods, FUTR3D [8] defines the 3D reference points as queries and directly samples the features from the coordinates of projected planes. TransFusion [1] follows a two-stage pipeline. The proposals are generated by LiDAR features and further refined by querying the image features.

2.4. Transformer-based Object Detection

The pioneering work DETR [3] proposes a transformer-based detector paradigm without any hand-craft components, and has achieved state-of-the-arts in both 2D and 3D detection [57, 6, 26, 30]. However, DETR-like methods usually suffer from the slow convergence. To this end, many works [60, 56, 28, 21, 57, 5, 18] are proposed to improve the training efficiency from various aspects. Other improvements in 2D detection mainly focus on modifying the transformer layers [60, 56], designing informative object queries [28, 21, 57], or exploring the label assignment mechanism [5, 18]. Deformable DETR [60] proposes the deformable attention, which only attends to sampling points of local regions. SAM-DETR [56] presents a semantic aligner between object queries and encoded features to accelerate the matching process. To alleviate the instability of bipartite matching, DAB-DETR [28] formulates the object queries as dynamic anchor boxes, while DN-DETR [21] auxiliarly reconstructs the ground-truths from the noisy ones. Based on them, DINO [57] further improves the denoising anchor boxes via a contrastive way.

3. Method

The overall architecture of the proposed CMT is illustrated in Fig. 3. Multi-view images and LiDAR points are fed into two individual backbones to extract multi-modal tokens. The 3D coordinates are encoded into the multi-modal tokens by the coordinates encoding. The queries from the position-guided query generator are used to interact with the multi-modal tokens in transformer decoder and then predict the object class as well as the 3D bounding boxes. The whole framework is learned in a fully end-to-end manner and the LiDAR backbone is trained from scratch without pretraining.

3.1. Coordinates Encoding Module

The coordinates encoding module (CEM) is used to encode the 3D position information into multi-modal tokens. It generates both the camera and BEV position encodings (PEs), which are added to image tokens and point cloud tokens respectively. With the help of CEM, multi-modal tokens can be implicitly aligned in 3D space.

Let \( P(u, v) \) be the 3D points set corresponding to the feature map \( F(u, v) \) of different modalities. Here \((u, v)\) indicates the coordinate in the feature map. Specifically, \( F \) is the image feature for camera while BEV feature for LiDAR. Suppose the output position embedding of CEM is \( \Gamma(u, v) \), its calculation can be formulated as:

\[
\Gamma(u, v) = \psi(P(u, v))
\]

where \( \psi \) is a multi-layer perception (MLP) layer.

**CE for Images.** Since the image is captured from a perspective view, each pixel can be seen as an epipolar line in 3D space. Inspired by PETR [29], for each image, we encode a set of points in camera frustum space to perform the coordinates encoding. Given the image feature \( F_{im} \), each pixel can be formulated as a series of points \( \{p_k(u, v) = (u + d_k, v + d_k, d_k, 1)^T, k = 1, 2, \ldots, d\} \) in the camera frustum coordinates. Here, \( d \) is the number of points sampled along the depth axis. The corresponding 3D points can be calculated by:

\[
p_{im}^k(u, v) = T_{ci}^d K_i^{-1} p_k(u, v)
\]

where \( T_{ci} \in R^{4\times4} \) is the transformation matrix from the \( i \)-th camera coordinate to the LiDAR coordinate. \( K_i \in 4 \times 4 \)
formed into 3D world space by linear transformation:

\[ p_k^{im}(u, v, k = 1, 2, ..., d) \]

where \( p_k \) is the intrinsic matrix of \( i \)-th camera.

The position encoding of pixel \((u, v)\) for image is formulated as:

\[ \Gamma_{im}(u, v) = \psi_{im}(\{p_k^{im}(u, v, k = 1, 2, ..., d)\}) \]

**CE for Point Clouds.** We choose VoxelNet [51, 58] or PointPillar [19] as backbone to encode the point cloud tokens \( F_{pc} \). Intuitively, the point set \( P \) in Eq. (1) can be sampled along the Z-axis. Suppose \((u, v)\) is the coordinates in BEV feature map, the sampled point set is then \( p_k(u, v) = (u, v, h_k, 1)^T \), where \( h_k \) indicates the height of \( k \)-th points and \( h_0 = 0 \) as default. The corresponding 3D points of BEV feature map can be calculated by:

\[ p_k^{pc}(u, v) = (u * u_d, v * v_d, h_k, 1) \]

where \((u_d, v_d)\) is the size of each BEV feature grid. To simplify, we only sample one point along the height axis. It is equivalent to the 2D coordinate encoding in BEV space. The position embedding of point cloud can be obtained by:

\[ \Gamma_{pc}(u, v) = \psi_{pc}(\{p_k^{pc}(u, v, k = 1, 2, ..., h)\}) \]

**3.2. Position-guided Query Generator**

Following Anchor-DETR [48] and PETR [29], we firstly initialize the queries with \( n \) anchor points \( A = \{a_i = (x_{ai}, y_{ai}, z_{ai}), i = 1, 2, ..., n\} \) sampled from uniform distribution between \([0, 1]\). Then these anchor points are transformed into 3D world space by linear transformation:

\[
\begin{align*}
    a_{x,i} &= a_{x,i} + (x_{max} - x_{min}) + x_{min} \\
    a_{y,i} &= a_{y,i} + (y_{max} - y_{min}) + y_{min} \\
    a_{z,i} &= a_{z,i} + (z_{max} - z_{min}) + z_{min}
\end{align*}
\]

where \([x_{min}, y_{min}, z_{min}, x_{max}, y_{max}, z_{max}]\) is the region of interest (RoI) of 3D world space. After that, we project the 3D anchor points \( A \) to different modalities and encode the corresponding point sets by CEM. Then the positional embedding \( \Gamma_q \) of object queries can be generated by:

\[ \Gamma_q = \psi_{pc}(A_{pc}) + \psi_{im}(A_{im}) \]

where \( A_{pc} \) and \( A_{im} \) are the point set projected on BEV plane and image plane, respectively. The positional embedding \( \Gamma_q \) are further added with the query content embedding to generate the initial position-guided queries \( Q_0 \).

**3.3. Decoder and Loss**

As for the decoder, we follow the original transformer decoder in DETR [48] and use \( L \) decoder layers. For each decoder layer, the position-guided queries interact with the multi-modal tokens and update their representations. Two feed-forward networks (FFNs) are used to predict the 3D bounding boxes and the classes using updated queries. We formulate the prediction process of each decoder layer as follows:

\[ \hat{b}_i = \Psi_{reg}(Q_i), \hat{c}_i = \Psi_{cls}(Q_i), \]

where \( \Psi_{reg} \) and \( \Psi_{cls} \) respectively represent the FFN for regression and classification. \( Q_i \) is the the updated object queries of the \( i \)-th decoder layer.

For set prediction, the bipartite matching is applied for one-to-one assignment between predictions and ground-truths. We adopt the focal loss for classification and \( L1 \) loss for 3D bounding box regression:

\[ L(y, \hat{y}) = \omega_1 L_{cls}(c, \hat{c}) + \omega_2 L_{reg}(b, \hat{b}) \]
Table 1: Performance comparison on the nuScenes test set. “L” is LiDAR and “C” is camera.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Modality</th>
<th>NDS↑ mAP↑</th>
<th>mATE↓ mASE↓ mAOE↓ mAVE↓ mAAE↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEVDet [17]</td>
<td>C</td>
<td>0.488 0.424 0.524 0.242 0.373 0.950 0.148</td>
<td></td>
</tr>
<tr>
<td>DETR3D [47]</td>
<td>C</td>
<td>0.479 0.412 0.641 0.255 0.394 0.845 0.133</td>
<td></td>
</tr>
<tr>
<td>PETR [29]</td>
<td>C</td>
<td>0.504 0.441 0.593 0.249 0.383 0.808 0.132</td>
<td></td>
</tr>
<tr>
<td>CenterPoint [54]</td>
<td>L</td>
<td>0.673 0.603 0.262 0.239 0.361 0.950 0.148</td>
<td></td>
</tr>
<tr>
<td>UVTR [22]</td>
<td>L</td>
<td>0.697 0.639 0.302 0.246 0.350 0.207 0.123</td>
<td></td>
</tr>
<tr>
<td>TransFusion [1]</td>
<td>L</td>
<td>0.702 0.655 0.256 0.240 0.351 0.278 0.129</td>
<td></td>
</tr>
<tr>
<td>PointPainting [42]</td>
<td>LC</td>
<td>0.610 0.541 0.380 0.260 0.541 0.293 0.131</td>
<td></td>
</tr>
<tr>
<td>PointAugmenting [43]</td>
<td>LC</td>
<td>0.711 0.668 0.253 0.235 0.354 0.266 0.123</td>
<td></td>
</tr>
<tr>
<td>MVP [7]</td>
<td>LC</td>
<td>0.705 0.664 0.263 0.238 0.321 0.313 0.134</td>
<td></td>
</tr>
<tr>
<td>FusionPainting [50]</td>
<td>LC</td>
<td>0.716 0.681 0.256 0.236 0.346 0.274 0.132</td>
<td></td>
</tr>
<tr>
<td>UVTR [22]</td>
<td>LC</td>
<td>0.711 0.671 0.306 0.245 0.351 0.225 0.124</td>
<td></td>
</tr>
<tr>
<td>TransFusion [1]</td>
<td>LC</td>
<td>0.717 0.689 0.259 0.243 0.359 0.288 0.127</td>
<td></td>
</tr>
<tr>
<td>DeepInteration [52]</td>
<td>LC</td>
<td>0.729 0.702 0.261 0.239 0.329 0.260 0.134</td>
<td></td>
</tr>
<tr>
<td>CMT-C</td>
<td>C</td>
<td>0.481 0.429 0.616 0.248 0.415 0.904 0.147</td>
<td></td>
</tr>
<tr>
<td>CMT-L</td>
<td>L</td>
<td>0.701 0.653 0.286 0.243 0.356 0.238 0.125</td>
<td></td>
</tr>
<tr>
<td>CMT</td>
<td>LC</td>
<td>0.741 0.720 0.279 0.235 0.308 0.259 0.112</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Performance comparison on the nuScenes val set. “L” is LiDAR and “C” is camera.

<table>
<thead>
<tr>
<th>Methods</th>
<th>modality</th>
<th>NDS↑ mAP↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUTR3D [8]</td>
<td>L</td>
<td>0.655 0.593</td>
</tr>
<tr>
<td>UVTR [22]</td>
<td>L</td>
<td>0.676 0.608</td>
</tr>
<tr>
<td>TransFusion [1]</td>
<td>L</td>
<td>0.701 0.651</td>
</tr>
<tr>
<td>FUTR3D [8]</td>
<td>LC</td>
<td>0.683 0.645</td>
</tr>
<tr>
<td>UVTR [22]</td>
<td>LC</td>
<td>0.702 0.654</td>
</tr>
<tr>
<td>TransFusion [1]</td>
<td>LC</td>
<td>0.713 0.675</td>
</tr>
<tr>
<td>BEVFusion [31]</td>
<td>LC</td>
<td>0.714 0.685</td>
</tr>
<tr>
<td>DeepInteration [52]</td>
<td>LC</td>
<td>0.726 0.699</td>
</tr>
<tr>
<td>CMT-C</td>
<td>C</td>
<td>0.460 0.406</td>
</tr>
<tr>
<td>CMT-L</td>
<td>L</td>
<td>0.686 0.624</td>
</tr>
<tr>
<td>CMT</td>
<td>LC</td>
<td>0.729 0.703</td>
</tr>
</tbody>
</table>

where $\omega_1$ and $\omega_2$ are the hyper-parameter to balance the two loss terms. Note that for positive and negative queries in query denoising, the loss is calculated in the same way.

3.4. Masked-Modal Training for Robustness

Security is the most important concern for autonomous driving systems. An ideal system requires solid performance even if part of them fails, as well as not relying on any input of a specific modality. Recently, BEVFusion [27] has explored the robustness of LiDAR sensor failure. However, the exploration is limited to restricted scan range and model need be retrained. In this paper, we try more extreme failures, including single camera miss, camera miss and LiDAR miss, as shown in Fig. 4. It is consistent with the actual scene and ensures the safety of autonomous driving.

Figure 4: We analyze the system robustness of CMT at test period under three simulated sensor errors: (a) single camera miss, (b) all camera miss and (c) LiDAR miss.

To improve the robustness of the model, we propose a training strategy, called masked-modal training. In training process, we randomly use only a single modality for training, such as camera or LiDAR, with the ratio of $\eta_1$ and $\eta_2$. This strategy ensures that the model are fully trained with both single modal and multi-modal. Then the model can be tested with single modal or multi-modal, without modifying the model weight. The experimental results show that masked-modal training will not affect the performance of our fusion model. Even if LiDAR is damaged, it can still achieve similar performance compared to the SoTA vision-based 3D detectors [29, 17] (see Tab. 3-4).
Table 3: Quantitative results on the nuScenes val with LiDAR or camera miss. With the masked-modal training, the efficiency and robustness of our CMT is significantly improved, especially when the LiDAR camera is missed.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Vanilla training CMT</th>
<th>Masked-modal training CMT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>only LiDAR</td>
<td>only Cams</td>
</tr>
<tr>
<td>NDS ↑</td>
<td>0.726 0.603 0.073</td>
<td>0.729 (↑0.3%) 0.681 (↑7.8%) 0.447 (↑37.4%)</td>
</tr>
<tr>
<td>mAP ↑</td>
<td>0.691 0.487 0.000</td>
<td>0.703 (↑1.2%) 0.617 (↑13.0%) 0.383 (↑38.3%)</td>
</tr>
</tbody>
</table>

Table 4: NDS/mAP comparison on nuScenes val with sensor miss. BEVFusion is trained with mask-modal strategy. * means our reproduced result.

<table>
<thead>
<tr>
<th>Model</th>
<th>Both</th>
<th>Test modal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>only LiDAR</td>
<td>only Cams</td>
</tr>
<tr>
<td>TransFusion[1]</td>
<td>0.71/0.67</td>
<td>0.70/0.65 None</td>
</tr>
<tr>
<td>BEVFusion[31]*</td>
<td>0.72/0.68</td>
<td>0.68/0.63 0.40/0.32</td>
</tr>
<tr>
<td>CMT</td>
<td>0.73/0.70</td>
<td>0.68/0.62 0.45/0.38</td>
</tr>
</tbody>
</table>

Table 5: CDS/AP comparison on Argoverse2 val set. “L” is LiDAR and “C” is camera.

<table>
<thead>
<tr>
<th>Model</th>
<th>Modality</th>
<th>AP</th>
<th>CDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>VoxelNeXt[11]</td>
<td>L</td>
<td>0.307</td>
<td>-</td>
</tr>
<tr>
<td>FSF[23]</td>
<td>LC</td>
<td>0.332</td>
<td>0.255</td>
</tr>
<tr>
<td>CMT</td>
<td>LC</td>
<td>0.361</td>
<td>0.278</td>
</tr>
</tbody>
</table>

3.5. Discussion

CMT shares similar motivation with FUTR3D [8] on the end-to-end modeling. However, both the method and its effectiveness are totally different. FUTR3D repeatedly samples the corresponding features from each modal and then performs the cross-modal fusion. CMT conducts the position encoding for both multi-view images and point clouds, which are simply added with corresponding modal tokens, removing the repeated projection and sampling processes. It keeps more end-to-end spirits in original DETR framework. Moreover, CMT achieves much better performance compared to the FUTR3D (see comparison in Tab. 1), showing its superior effectiveness. We think CMT provides a better end-to-end solution for multi-modal object detection.

4. Experiments

4.1. Datasets and Metrics

We evaluate our method on open datasets, including nuScenes [2] and Argoverse 2 [49].

NuScenes [2] is a large-scale multi-modal dataset, which is composed of data from 6 cameras, 1 LiDAR and 5 radars. The dataset has 1000 scenes totally and is divided into 700/150/150 scenes as train/validation/test sets, respectively. Each scene has 20s video frames with 12 FPS. 3D bounding boxes are annotated every 0.5s. We only use these key frames. In each frame, nuScenes provides images from six cameras. NuScenes provides a 32-beam LiDAR with 20 FPS. The key frames are also annotated every 0.5s, the same as cameras. We follow the common practice to transform the points from the past 9 frames to the current frame for training and evaluation. We follow the nuScenes official metrics.

We report the nuScenes Detection Score (NDS), mean Average Precision (mAP), mean Average Translation Error (mATE), mean Average Scale Error (mASE), mean Average Orientation Error(mAOE), mean Average Velocity Error (mAVE) and mean Average Attribute Error (mAAE).

Argoverse 2(AV2) [49] contains 1000 sequences in total, 700/150/150 for train/validation/test similar as nuScenes. AV2 provides a long perceptron range up to 200 meters, covering an area of 400m × 400m, which is much larger than nuScenes. We report mean Average Precision(mAP), Composite Detection Score(CDS).

4.2. Implementation Details

We use ResNet[16] or VoVNet[20] as image backbone to extract the 2D image features. The C5 feature is upsampled and fused with C4 feature to produce P4 feature. We use VoxelNet [58] or PointPillars [19] as the backbone to extract the point-cloud features. All the feature dimension is set to 256, including the LiDAR feature, image feature and query embedding. Six decoder layers are adopted in transformer decoder.

Our model is trained with the batch size of 16 on 8 A100 GPUs. It is trained for total 20 epochs with CBGS[59]. We adopt the AdamW[32] optimizer for optimization. The initial learning rate is 1.0×10^{-4} and we follow the cycle learning rate policy[40]. The mask ratios η1 and η2 are both set to 0.25 for masked-modal training. The GT sample augmentation is employed for the first 15 epochs and closed for the rest epochs. As for the loss weights, we follow the default setting in DETR3D [47] and set the ω1 and ω2 to 2.0 and 0.25, respectively. For fast convergence, we introduce the point-based query denoising strategy based on DNDETR [21]. Different from it, we generate the noisy anchor points by center shifting since the box scale is not that important in 3D object detection.
4.3. State-of-the-Art Comparison

As shown in Tab. 1, CMT achieves state-of-the-art performance compared to existing methods on nuScenes test set. Our LiDAR-only baseline, named CMT-L, achieves 70.1% NDS, which is a nearly SoTA performance among all existing LiDAR-only methods. Our multi-modal CMT achieves 74.1% NDS and 72.0% mAP, outperforming all existing SoTA approaches, such as BEVFusion [31] and DeepInteration [52]. We also compare the performance with other SoTA methods on nuScenes val set (see Tab. 2). It shows that our proposed CMT with multi-modal fusion outperforms the BEVFusion by 1.8% mAP. CMT introduces large performance improvements compared to our LiDAR-only CMT-L by 4.0%/6.7% and 4.3%/7.9% NDS/mAP on test and validation set, respectively. In comparison, TransFusion only brings 1.5%/3.4% NDS/mAP on test set, compared to the LiDAR-only TransFusion. It shows that the multi-view images bring much more complementary information to the point clouds in CMT framework. We think the end-to-end modeling of CMT relatively improves the importance of image tokens. Fig. 5 shows some qualitative detection results on the nuScenes test set.

On AV2 dataset, CMT also outperforms existing SoTA methods, including VoxelNeXt[11] and FSF[23], as shown 4.4. Strong Robustness

We evaluate the robustness of our framework under various harsh environments, including LiDAR miss and camera miss. Tab. 3 shows the results when the sensor miss occurs, by simulating the scenarios of any modality totally broken. The performance is compared between the vanilla training and masked-modal training. It validates the effect of masked-modal training. Note that the model are only trained with multi-modality and evaluated without any fine-tune process. With vanilla training, the model fails to predict anything meaningful (only Cams with mAP=0) when LiDAR is missing. With masked-modal training, the absence of LiDAR or camera modalities lead to 4.8% and 28.2% NDS drop compared to CMT, respectively. It is observed that losing one modality still remains similar results compared to single-modal training settings. It overcomes the drawback that multi-modal method usually rely on one major modality and performance would degrade significantly if losing the major modality. Especially, for the case of LiDAR missing, the performance is still comparable to the SoTA camera-only method PETR [29], validating the strong robustness of our method. We further evaluate the performance of TransFusion and BEVFusion under sensor miss (see Tab. 4). TransFusion fails to work when LiDAR is missing due to the two-stage design. With the masked-modal training, BEVFusion achieves the decent performance (40% NDS and 32% mAP), while showing large inferiority compared to CMT.

Moreover, we also investigate the case when any one of cameras fails. Experimental result shows slight performance drop, indicating the tolerable to single camera miss
of our method. Six sensors brings an average decrease of 0.7% NDS, no more than 1% performance of the oracle version. The front and back sensor relatively play the important role among camera sensors, with 1.1% and 0.8% decrease respectively, due to their distant or large field of view. Compared to the camera-only setting, our multi-modal framework facilitate the compensation between LiDAR and image domains, thus presenting a robust performance.

4.5. Ablation Study

We present ablation studies in Tab. 6. All experiments are conducted for 20 epochs without CBGS[34]. We first ablate the effect of Im PE and PC PE on the generation of position-guided queries. It shows that removing PC PE introduces a 7.4%/8.70% NDS/mAP performance drop, which is much larger than the drop of removing Im PE 0.4%/1.5%. Next, we explore the effectiveness of point-based query denoising (PQD) introduced in Sec. 4.2. We can easily find that PQD can greatly improve the overall performance by 4.3%/5.7% NDS/mAP. With PQD, the training convergence can be boosted, which is similar to the practice in DN-DETR [21]. Further, we also illustrate the effect of scaling up the CMT model as well as the input size. Overall, CMT can benefit from the scaling model size. Interestingly, we find increasing the voxel number (smaller voxel size) and image size achieves similar improvements $\approx 1.5\%$ in NDS. While scaling the image size increases more mAP than the voxel number(+3.2% vs. +1.7%). When increasing the image size from $800 \times 320$ to $1600 \times 640$, we find the performance improvements are mainly from these small objects, such as pedestrian and motorcycle. We also conduct experiments on replacing image and LiDAR backbones, we use VoV-99[20] and ResNet[16] as our image backbones. Experiments show that our proposed CMT can benefit from larger backbones. For image, VoV-99 backbone achieves the best result and outperforms the ResNet-50 by 1.1%/1.8% in NDS/mAP. While for LiDAR, VoxelNet outperforms the PointPillar by 4.1%/4.3% in NDS/mAP.

4.6. Analysis

For better understanding on querying from multi-modal tokens, we visualize the attention map of cross-attention on the multi-view images (see Fig. 6). We can clearly find that the attention maps have higher response on the regions that includes foreground objects. It shows that our method can implicitly achieve the cross-modal interaction. We visualize the initial anchor points and the center points of predictions. Most anchor points focus on the closest foreground objects. After the interaction with multi-modal tokens in the transformer decoder, anchor points are updated and gradually access the accurate center points.

5. Conclusions

In this paper, we propose a fully end-to-end framework for multi-modal 3D object detection. It implicitly encodes the 3D coordinates into the tokens of images and point clouds. With the coordinates encoding, the simple yet effective DETR pipeline can be adopted for multi-modal fusion and end-to-end learning. With masked-modal training, our multi-modal detector can be learned with strong robustness, even if one of multi-modalities are missed. We hope such a simple pipeline design could provide more insights on the end-to-end 3D object detection.
Acknowledgements: This research was supported by National Key R&D Program of China (No. 2017YFA0700800) and Beijing Academy of Artificial Intelligence (BAAI).

References


