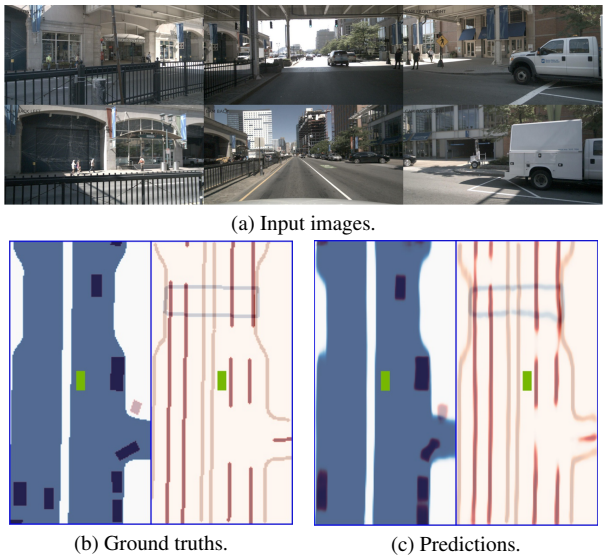
Scene Representation in Bird's-Eye View from Surrounding Cameras with Transformers

Yun Zhao Yu Zhang Zhan Gong Hong Zhu
Department of AI and HPC Inspur Electronic Information Industry Co., Ltd
Beijing, China
{zhaoyun02, yu.zhang01, gongzhan01, zhuhongbj}@inspur.com

Abstract

Scene representation in the bird's-eye-view (BEV) coordinate frame provides a succinct and effective way to understand surrounding environments for autonomous vehicles and robotics. In this work, we present an end-to-end architecture to generate the BEV representation from surrounding cameras. To generate the BEV representation, we propose a transformer-based encoder-decoder structure to translate the image features from different cameras into the BEV frame, which takes advantage of the context information in the individual image and the relationship between images in different views. We perform multiple semantic segmentation tasks using the BEV features. Experimental results show that our model outperforms the competitive baseline [20], which demonstrates the effectiveness and efficiency of our method.



(a) Input images.

(b) Ground truths. (c) Predictions.

Figure 1. (a) Images from surrounding cameras. (b) Ground truths of semantic segmentations in the BEV frame. Left sub-figure: vehicle segmentation (dark blue or orange), road segmentation (blue). Right sub-figure: lane boundary (orange), lane divider (dark red), pedestrian crossing (blue). (c) Predictions of our model. The green rectangle represents the ego vehicle.

1. Introduction

Scene perception is a fundamental task for autonomous driving and robotics, which builds an expressive representation of the surrounding scene that captures the geometry and layout of the static world and the location and dimension of the dynamic agents. Most competitive 3D object detectors rely on LiDAR data as it can provide a distance measurement to the environment. However, the points cloud is low resolution with little texture information [11, 27], and the LiDAR devices are expensive. On the contrary, monocular cameras are cheap and capable of capturing abundant contextual information, making them extensively researched and applied in both academic institutions and companies. Instead of depicting the 3D environment entirely, a succinct manner represents the surrounding scene in the form of a birds-eye-view (BEV) map [20, 22], which captures the spatial layout of the environment adequately and provides convenience for the downstream motion planning task [32].

To construct BEV maps from multi-view images, a straightforward and practical strategy is performing the perception tasks (such as 3D object detection, road segmentation, lane detection, depth estimation, *etc.*) on every single image individually and transforming the results from images in different views into a unified BEV frame according to the camera parameters and geometric constraints. This multistage processing pipeline takes advantage of the outstanding works on image-based tasks, in which the 2D detection results are robust against the configuration of the cameras [20]. However, this result-based transformation and fusion paradigm is weak in exploring the relationship between different cameras, which creates confusion during fusing results from different images.

Recently, many approaches [7,20] have followed an end-to-end pipeline, which transforms the image-base features into the BEV frame and directly performs perception tasks on the BEV feature maps. Inferring the road elements directly on the BEV feature map containing the spatial configurations can gain robust results. In these methods, the critical component is transforming the image features into the BEV frame and fusing the BEV features from different cameras. An intuitive approach for feature transformation is projecting the image feature in each pixel into the corresponding location in the BEV frame using the camera parameters. The independent projection ignores the contextual information for locating the image features into the BEV frame. So, some methods [7, 18] introduce a multi-layer perceptron (MLP) to use the whole features in an image. However, they miss the contextual information between neighboring cameras, and the relationship between the features in the overlap region of nearby cameras is significant for feature transformation. On the other hand, the following addition operation is generally implemented to integrate projected features from different images into the same BEV pixel during feature fusion. Despite efficiency, this simple fusion mode is limited to represent the relationships between different cameras.

In this work, we present an end-to-end framework that extracts the BEV representation of a scene from surrounding cameras. A CNN-based image encoder is utilized to generate features in the image view. To make use of the contextual information sufficiently, we introduce a transformer-based encoder-decoder structure to *translate* the image features from different cameras into the BEV frame. Instead of using the intrinsics and extrinsics of the cameras, the transformer-based module fuses and transforms the image features based on the self-/cross-attention mechanism with the position encodings. The following BEV encoder is performed on the BEV features to encode the spatial information further. To demonstrate the effectiveness of the BEV representation, we implement different semantic segmentation tasks using several separate CNN-based task branches.

Our main contributions are summarised as follows:

- We propose an end-to-end structure to generate the BEV features from multiple images captured by surrounding cameras, which is convenient to perform perception tasks and following planning tasks in a reasonable end-to-end manner.
- We propose a transformer-based encoder-decoder framework to transform the image features from different cameras into the BEV features, which can effectively explore the relationships between the image features from surrounding cameras and fully take advantage of the contextual information to generate the BEV features.
- We perform multiply semantic segmentation tasks and demonstrate the effectiveness and efficiency of our model.

2. Related Work

Monocular 3D Object Detection. Monocular 3D object detectors predict 3-dimensional rotated bounding boxes from monocular images. This task faces significant challenges as regressing 3D information from only image appearances is an ill-posed problem. Recently, many approaches have been proposed in recent years and achieved gradually increase both in terms of detection accuracy and inference speed. Here, we briefly present the monocular 3D detection approaches. For more details, we recommend that researchers refer to the survey [15].

Most monocular 3D detectors [14,17,28,29] share a similar paradigm with 2D detection models, which first generate 2D object information (including 2D locations, dimensions, orientations, depths, *etc*.) and then project the 2D results into the 3D space. The geometric relationship between 2D image and 3D space is always utilized to assist monocular 3D detection [1, 10]. M3D-RPN [1] introduces shared 2D and 3D anchors using the information between 2D scale and 3D depth and proposes depth-aware convolutions to generate location-specific features. Instead of regressing the 3D bounding boxes directly, RTM3D [10] performs the key-point detection and recovers the 3D bounding box using the key-points and geometric constraints. Besides geometric constraints, shape information is also introduced to alleviate the projection ambiguity [4, 9, 12, 16]. Benefited from the rapid development of 2D object detection algorithms, these direct monocular 3D detection algorithms have made remarkable progress.

Recently, a pseudo-LiDAR pipeline [30, 31] was proposed, which performs monocular depth estimation and LiDAR-based 3D detection separately. These methods generate the estimated depth map and transform it into a 3D point cloud, named pseudo-LiDAR. Then, the LiDAR-based detection approaches are employed to extract the 3D results from the pseudo-LiDAR data. The outstanding performance of these detectors shows the importance of the spatial features for 3D object detection and the reasonability of predicting the results in the 3D coordinate frame.

Another category of monocular 3D detectors [21,23] performs 3D detection on the BEV features generated by projecting 2D image features into 3D voxel features and collapsing the 3D voxel features along the vertical dimension. OFTNet [23] generates the 3D feature for each voxel by average pooling over the area of the 2D image feature corresponding to the voxel cube. CaDDN [21] follows the project mechanism in [20]. These features achieve remarkable performance by combining the expressive image features and the spatial configuration of the scenes.

BEV representation from surrounding cameras. Recently, a growing number of approaches have been proposed to represent the surrounding environment from a group of cameras. A straightforward approach [5] is employing inverse perspective mapping (IPM) to map front-view image or semantic segmentation results onto the ground plane with a homography. The effectiveness of this approach stands on the flatness assumption, while the object in the 3D world always violates this assumption. Other approaches [7, 20, 22] follow a similar framework that generates image features in perspective and transforms them into BEV. PON [22] collapses the vertical dimension of the image feature map into bottleneck features and applies 1D convolution along the horizontal axis of the bottleneck features. The resulting features are reshaped into a given shape and re-sampled into the BEV frame using camera parameters. VPN [18] flattens the image-view feature map and employs multi-layer perceptron (MLP) to explore the relationship between any two-pixel positions in flattened image-view feature map and flattened BEV feature map. In LSS [20], the depth space is discretized into multiple bins, and the detector estimates the probability that the feature pixel locates in the bins. Then, each feature pixel is scattered into the frustum space in the camera coordinate according to the depth of each bin and weighted by the corresponding probability value. The frustum features in the camera coordinate are transformed into the 3D world coordinate frame to generate the voxel features. The generated features splat onto the reference plane to get the BEV features. We employ a similar pipeline to generate the BEV features by transforming the image features into the BEV space. Differently, we utilize a transformer-based encoder-decoder structure to *translate* the image features to the BEV features directly, which effectively integrates the contextual and geometric information.

3. The Proposed Method

The overview framework of our model is shown in Fig. 2. Given a set of images $\{\mathbf{I}_k \in \mathbb{R}^{3\times H\times W} | k = 1, \ldots, N_c\}$ from N_c surrounding cameras where H and W represent the height and width of the images, our model generates a rasterized representation of the scene in the BEV coordinate frame. Sequentially, the BEV features are utilized for the diverse tasks.

3.1. Camera-BEV Transformation (CBTR)

A camera-BEV feature transformation (CBTR) architecture is designed to gain the effective BEV representation of scenes, which fuses the deep features from each monocular camera and transforms them into the BEV space. The overall architecture follows the general encoder-decoder structure [3, 26]. The encoder component fuses the features from different cameras and strengthens the expression of image features, and the decoder component generates the BEV feature representation based on the image features from the encoder module. The detailed definition of architecture is depicted in Fig. 3.

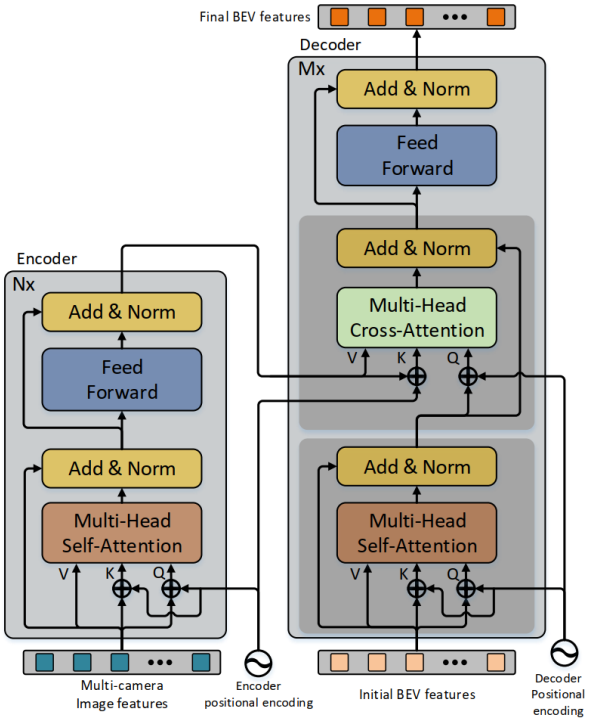


Figure 3. The pipeline of camera-BEV transformation that transforms the image-view features from different cameras into the BEV frame. N is the number of the encoder layer, M is the number of the decoder layer. The image-view features from surrounding cameras are flattened into the image feature sequence and fed into the transformer-based encoder. The decoder layer uses the BEV feature sequence from previous decoder layer and the image-view feature sequence from the encoder module to generate the final BEV features. Q, K and V represent the *query* sequence, *key* sequence, and *value* sequence for multi-head attention modules as described in [3].

Transformer Encoder. Before being fed to the encoder module, each input camera feature map is convolved with a $d \times 1 \times 1$ convolution, which changes the channel dimension of each feature map to d. All the feature maps from N_c camera images are gathered to form a $d \times N_c \times H_c \times W_c$ stacked feature map, where H_c and W_c denote the spatial dimensions of the image feature maps. As the encoder requires a sequence as input, we flatten the spatial dimensions of the stacked feature map to obtain a $d \times s_c$ input sequence, where $s_c = N_c H_c W_c$ represents the length of the input sequence.

The encoder contains N cascaded identical layers [26], where each layer consists of a multi-head self-attention module and a feed-forward network sequentially. After every sub-layer, a residual connection is applied, followed by

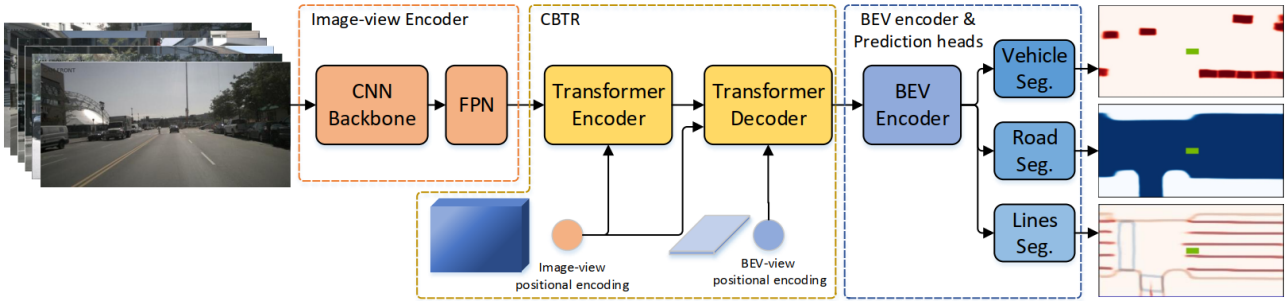


Figure 2. The overview of our model. A set of images from surrounding cameras are fed into the image-view encoder with a shared CNN structure to generate image-view features. Then, the camera-bev transformation (CBTR) module is utilized to fuse the image-view features from different cameras and transform them into the BEV frame. An CNN-based BEV encoder is introduced to strengthen the BEV features. Finally, the BEV features are employed for different road elements detection tasks.

a layer normalization. Since spatial clues are essential for camera-BEV feature transformation, we generate positional encodings for the image feature maps and add them to the input image features in each encoder layer.

Transformer Decoder. The decoder module consists of M identical layers, where each layer is divided into three parts: self-attention module, cross-attention module, and feed-forward network, as shown in Fig. 3. Each encoder layer receives the BEV feature sequence from the previous layer, the BEV feature positional encodings, the output image features from the encoder, the camera feature positional encodings, and products the improved BEV features through the cascaded multi-head self-attention layer, multi-head cross-attention layer, and feed-forward layer.

The self-attention layer accepts the input BEV feature sequence and corresponding positional encodings, where the BEV feature sequence is treated as the *values*, and the addition of the BEV feature sequence and corresponding positional encodings is regarded as *keys* and *queries* as described in [3]. The initial BEV feature map is constructed according to the range and resolution of BEV space, which is of $H_{bev} \times W_{bev}$ spatial shape and d-dimension channel. Then, it is flatten into a $d \times s_{bev}$ sequence, where $s_{bev} = H_{bev}W_{bev}$ denotes the sequence length.

The cross-attention layer transforms the camera features into the BEV frame. It receives the encoded camera features as *values*, the addition of encoded camera features and corresponding camera feature positional encodings as *keys*, and the addition of the processed BEV features and the BEV positional encodings as *queries*. The *queries* are processed in parallel. The output of the cross-attention layer is fed to a fully connected feed-forward network. Finally, the improved BEV features from the last encoder layer are reshaped to the pre-designed spatial shape for the following networks.

A simple and efficient way to initialize the BEV feature map is setting all the values to zero. In this case, the self-attention layer can be omitted in the first decoder layer. A more effective method to initialize the BEV feature map is transforming the camera features according to the intrinsic and extrinsic matrices of the cameras and the BEV spatial setting as LS [20]. However, it significantly reduces the computation speed.

Positional Encodings. There are two kinds of positional encodings: camera feature positional encodings and BEV feature positional encodings, among which camera feature positional encodings are utilized at attention layers in both the encoder and decoder layers, as shown in Fig. 3. To encode positional information of multiple image features, we generalize the positional encodings in the 2D case [19] to the 3D case. Specifically, we use sine and cosine functions of different frequencies independently for spatial and index coordinates of the stacked camera feature map. Then, the positional encodings in different coordinates are concatenated to form the final positional encodings.

The fixed positional encodings in the 3D case described above is a simple implementation for multiple camera feature maps. It is a suboptimal method for representing positional information of camera features, as it cannot express the positional relations between covered regions of nearby cameras. However, it works well on Nuscenes [2] dataset. An important reason is that the monocular images from neighboring cameras in the Nuscenes dataset contain few overlapping areas.

Computation complexity and Storage consumption. We discuss the computation complexity and storage consumption in self- and cross- attention mechanisms here to guide the selection of the hyper-parameters.

During encoding, the computational complexity of the multi-head self-attention (MSA) module based on the stacked camera feature map with spatial dimensions $N_c \times H_c \times W_c$ is:

$$\Omega(MSA) = 4(N_c H_c W_c) \cdot d^2 + 2(N_c H_c W_c)^2 \cdot d. \quad (1)$$

The former term is quadratic to the feature dimension, and

the latter is quadratic to the length of the input feature sequence. What's worse, the intermediate product of query and key matrices (QK^T in [26]) consumes $\mathcal{O}((N_cH_cW_c)^2)$ memories for each head. It becomes a thorny issue for training when the length of the flattened camera feature map is too large.

The self-attention module in the decoder layer faces a similar dilemma as in encoder layers when the length of the flattened BEV features $H_{bev} \times W_{bev}$ is too large. Meanwhile, the computational complexity of the multi-head cross-attention (MCA) module in the model is:

$$\begin{aligned}\Omega(MCA) =& 2(N_cH_cW_c) \cdot d^2 + 2(H_{bev}W_{bev}) \cdot d^2 \\ &+ 2(N_cH_cW_cH_{bev}W_{bev}) \cdot d. \end{aligned} \tag{2}$$

The intermediate product of query and key matrices consumes $\mathcal{O}(N_cH_cW_cH_{bev}W_{bev})$ memories for each head. Obviously, the resolution of the BEV feature map is a critical factor for computation speed and memory cost.

To alleviate these issues, we appropriately reduce the spatial resolution of the feature maps and the number of the encoder and decoder layers. During encoding, we reduce the spatial resolution of the image feature map by shrinking the original input images. Meanwhile, considering the CNN-based image-view encoder can generate expressive features, we employ a few encoder layer in our model and measure the effect of the encoder number in Tab. 1. During decoding, considering the important and unique role of the decoder modules for transforming the image-view features, we reserve relatively sufficient encoder layers but reduce the spatial size of the BEV feature map.

3.2. Other Network Structures

In this section, we introduce the details of other network structures in our propose model including the image-view encoder, the BEV encoder and the segmentation heads.

3.2.1 Image-view Encoder

The image-view encoder encodes each image to feature maps individually. We adopt a similar pipeline with the one described in [20]. In particular, an EfficientNet-B0 [25] pre-trained on ImageNet is used to extract image features in different scales. The final output image feature maps are $1/16\times$ input resolution. To combine multi-resolution features, the features with $1/32\times$ input resolution are up-sampled to $1/16\times$ input resolution by linear interpolation and fused with the features with $1/16\times$ input resolution by several convolution layers.

3.2.2 BEV Encoder

The BEV encoder encodes the transformed BEV features from the CBTR module. Since the feature map from the CBTR module is of $1/4\times$ final BEV resolution, a crucial function of the BEV encoder is up-scaling the BEV feature maps for the BEV prediction. Here, we adopt two modes: the light BEV encoder and the heavy BEV encoder. The light BEV encoder simply up-samples the transformed BEV features by linear interpolation. Generally, it needs a followed complicated task head to get good predictions. Differently, the heavy BEV encoder has a similar structure with the image-view encoder, which utilizes ResNet [6]-like module to construct and fuse the features. In this case, the task modules always share the BEV features with simple structures to gain the final results.

3.2.3 Segmentation Heads

Three separate neural network branches for semantic segmentation tasks are constructed upon the BEV feature map: vehicle segmentation, road segmentation, and lines segmentation. Each branch has an identical structure but different output channels in the last layer. The task-specific head utilizes ResNet-like blocks to extract features in different scales and combines these feature maps by up-sampling. Finally, the BEV feature map with the given BEV output spatial size is convolved with different numbers of convolution kernels to generate the output results. Specifically, the output channel for the three semantic segmentation is 1 for vehicle segmentation, 1 for road segmentation, and 3 for lines segmentation, respectively.

A light head with only final convolution with a shared heavy BEV encoder can gain more inference speed for the detector. However, we experimentally find that the separate heavy heads gain better performance. We argue that these three kinds of tasks focus on different kinds of features.

3.3. Training

We train the whole model with three segmentation tasks in the end-to-end manner. Each segmentation task is trained with binary cross-entropy with a positive weight (2.13 in our experments). The final loss used for training the model is:

$$\mathcal{L} = w_r \cdot \mathcal{L}_v + w_r \cdot \mathcal{L}_r + w_l \cdot \mathcal{L}_l, \tag{3}$$

where $\mathcal{L}_v$, $\mathcal{L}_r$, $\mathcal{L}_l$ represents the loss of vehicle segmentation, road segmentation, and lines segmentations, respectively. In following experiments, we empirically set the weights $w_v = 3$, $w_r = 1$ and $w_l = 3$

4. Experiments

4.1. Experiments Settings

Dataset. We evaluate our approach on the large-scale and popular dataset, nuScenes [2]. It consists of 1000 scenes captured from four locations in Boston and Singapore, each of 20 seconds in length, covering different conditions. All

Methods	Dec.-SA	nHeads	Enc.	Dec.	Vehicles	Road	Divider	Ped Crossing	Boundary	Runtime(ms)
A	×	2	0	3	52.63	90.78	45.13	29.28	43.88	34.64
B	×	2	0	6	53.15	90.99	44.89	29.39	43.73	36.98
C	×	2	1	3	53.07	**91.44**	44.95	**30.03**	44.22	35.66
D	×	4	1	3	53.18	90.64	45.94	29.65	44.17	39.17
E	×	2	3	3	53.11	91.01	44.70	29.53	44.26	39.56
F	✓	2	1	3	**53.85**	91.18	**45.90**	29.92	**44.93**	35.21

Table 1. Ablation study of CBTR. We evaluate different configurations on the nuScenes val subset with the IoU scores (%). Dec.-SA denotes the self-attention module in the decoder layers. nHeads represents the number of the heads in the multi-head attention module.

Cam. PE	BEV PE	Vehicles	Road	Divider	Ped Crossing	Boundary	Runtime(ms)
sine	learned	53.85	91.18	45.90	29.92	44.93	35.21
sine	sine	**54.01**	**91.34**	**46.65**	**31.47**	**46.03**	34.76
learned	learned	53.95	90.73	46.42	31.14	44.77	35.40
learned	sine	53.31	90.86	45.80	29.86	45.02	35.17

Table 2. IoU scores (%) for different positional encodings modes. The evaluation is implemented on the nuScenes val subset.

the scenes are officially split into $700/150/150$ scenes for training/validation/testing. Our experiments use the training and validation subset for training and testing. The images are captured from 6 surround-view cameras, which are well-calibrated and synchronized. The camera group provides a 360-degree view with a slight overlap between the neighboring cameras.

The nuScenes dataset provides annotated 3D objects and highly accurate human-annotated semantic maps of the relevant areas. For each scene, it first samples key-frames at 2HZ and annotates the object with a category, attributes, and a 3D bounding box. It also provides vectorized maps of different semantic classes and the pose of ego-vehicle.

Data Processing & Evaluation Metrics. The original input images in nuScenes with 1600×900 resolution are processed by random scaling and random rotating, and finally cropping to 640×320 resolution. We set the BEV region in x from -30 meters to 30 meters and in y from -15 meters to 15 meters, where the x axis denotes the forward direction and the y axis represents the left direction. The BEV region is rasterized into a grid map with 240×120 resolution, where each cell is of size 0.25 meters $\times$ 0.25 meters.

For vehicle segmentation, we obtain the ground truths by projecting annotated vehicle 3D bounding boxes into the BEV plane as [20], where the vehicle target denotes the meta-category *vehicle* in nuScenes including car, bus, trailer, motorcycle, *etc.* For road segmentation, we generate road ground truth BEV map by transforming the combination of the nuScenes map layer `road_segment` and `lane` into the ego frame. For lines segmentation, we follow the setting in [11] and extract three kinds of lines: lane boundary, lane divider, and pedestrian crossing. The width of each line is set to 2. The ground truths are illustrated in Fig. 1b.

During the evaluation, we employ intersection-over-union (IoU) as the evaluation metric for all segmentation tasks.

Training Details. The model is trained with AdamW [13] optimizer with one-cycle learning rate policy [24]. The max learning rate is $1e-3$, and the weight decay is $1e-6$. We train our model on A100 GPUs for 40 epochs.

4.2. Ablations

In the ablation analysis, we delve into how the components in CBTR and other modules in our model influent the final performance. The output BEV feature from CBTR is of $1/4\times$ dimension against the final BEV predictions.

4.2.1 CBTR

We evaluate the importance of the attention mechanism by changing the number of components in the CBTR. The results are shown in Tab. 1.

Interestingly, our model still gains good performance even without the encoder module. It indicates that the extracted image-based features are powerful enough to capture the appearance and spatial information. Comparing configuration *A* with *C,E*, increasing the number of encoder layer can improve the segmentation performance. However, too many encoder layers may decrease the performance, and more layers significantly increase the inference time and the memory cost.

The decoder module plays a crucial role in transforming the camera features. Nevertheless, comparing configuration *B* to *A*, too many decoder layers bring slight improvements

Module	BEV-Enc.	Task-Heads	Vehicles	Road	Divider	Ped Crossing	Boundary	Runtime(ms)
A	light	light	53.59	90.76	43.36	29.20	44.39	28.74
B	light	heavy	**54.01**	**91.34**	**46.65**	31.47	**46.03**	34.76
C	heavy	light	52.49	90.44	45.26	29.84	44.15	33.26
D	heavy	heavy	53.47	91.09	46.19	**31.80**	44.81	39.97

Table 3. IoU scores (%) of different combination modes for the BEV encoder and the task heads.

for segmentation. Taking a deeper insight into the decoder, we evaluate the importance of the self-attention mechanism in the encoder module. Comparing configuration *F* with *C*, we find the introduction of the self-attention mechanism in the encoder module can improve the segmentation performance, especially for vehicle segmentation. Meanwhile, the results in configuration *C* and *D* show that more heads in the attention mechanism bring minor improvements for accuracy but a considerable increase in inference time.

Effectiveness of positional encodings. Two kinds of positional encoding modes (fixed encoding and learned encoding) are utilized to generate image positional encodings and BEV positional encodings. The results are shown in Tab. 2. These two versions of the positional encoding method gain similar results, and the fixed sine positional encoding mode for both camera and BEV feature gain the best performance. We chose the fixed positional encoding method for our model's camera and BEV features.

4.2.2 BEV Encoder & Task Heads.

To study the impact of the network complexity for the BEV encoder and task head, we use two kinds of networks to implement the BEV encoder and task head, respectively. The light BEV-encoder refers to a simple up-sample operation, and the light task head denotes a two-layer CNNs module. The results are shown in Tab. 3.

As shown in configuration *A*, even with a light BEV encoder and a light task head, the model can achieve competitive results. This demonstrates that the BEV feature map from the CBTR module is an expressive representation for BEV perception tasks. Compared with the heavy BEV encoder, the heavy task head gains more improvements in segmentation performance, which shows that different segmentation tasks focus on different kinds of features. Heavy network for both BEV encoder and task head (*D*) achieves suboptimal results. The possible reason is that the learning in the BEV space falls into overfitting [8].

4.3. Comparison to other method

We compared our model with the state-of-the-art method LS [20]. We train the LS model under identical settings using the publicly available code. The comparison results are shown in Tab. 4. Our model gains better performance in segmentation accuracy and computation efficiency. It demonstrates that the transformer-based camera-BEV transformation can generate more expressive represents for scene understanding.

We show some instance results in Fig. 4. Our model gains more precise vehicle segmentation results than LS, especially for the vehicle that appeared in nearby cameras. Moreover, our models results are smooth because the construction of BEV features uses adequate contextual information.

5. Conclusions

In this paper, we present an end-to-end architecture to extract the BEV representations from surrounding cameras. We propose a transformer-based encoder-decoder structure to transform the image features into the BEV frame with the camera-based and BEV-based position encodings. This structure has capable of exploring the context information in each image and the relationship between different cameras. We perform multiple semantic segmentation tasks on the BEV features and gain excellent performance both on detection accuracy and computation speed that demonstrates the effectiveness of our model. In the future, we will optimize this transformer-based model in the design and computation factors (such as designing better position encodings manner, addressing the vast memory cost of the self-/cross-attention module, *etc*.) and introduce the temporal information to improve the performance in the current frame.

References

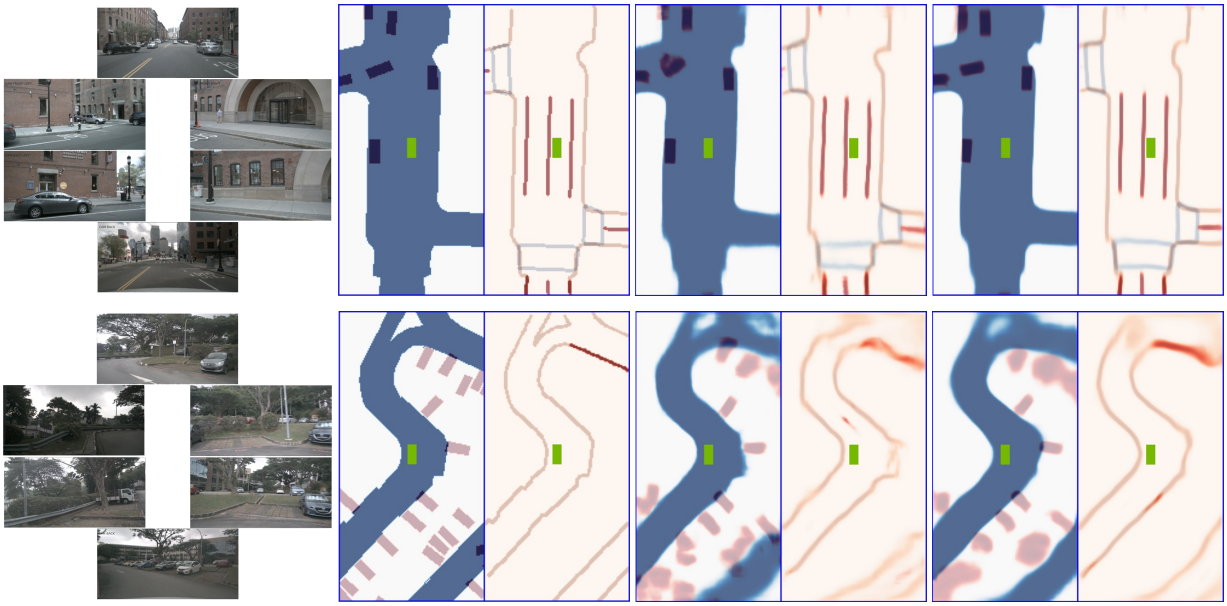
[1] Garrick Brazil and Xiaoming Liu. M3d-rpn: Monocular 3d region proposal network for object detection. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 2

[2] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 4, 5

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-

Module	Vehicles	Road	Divider	Ped Crossing	Boundary	Runtime(ms)
LS [20]	53.80	90.70	44.31	30.65	45.13	63.39
Our model	54.01	91.34	46.65	31.47	46.03	34.76

Table 4. Results comparison on the nuScenes val subset with the evaluation metric IoU(%).



(a) Input Images. (b) Ground Truths. (c) LS [20]. (d) Our model.

Figure 4. Qualitative comparison of the bird's-eye view predictions. (a) The input images from surrounding cameras. The top image is generates from the front camera. (b) The ground truth predictions. Left sub-figure: vehicle segmentation (dark blue or orange), road segmentation (blue). Right sub-figure: lane boundary (orange), lane divider (dark red), pedestrian crossing (blue). The green rectangle represents the ego vehicle. (c) The prediction results of LS [20] model. (d) The prediction results of our model.

end object detection with transformers. In *ECCV*, 2020. 3, 4

[4] Florian Chabot, Mohamed Chaouch, Jaonary Rabarisoa, Cline Teulire, and Thierry Chateau. Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[5] Liuyuan Deng, Ming Yang, Hao Li, Tianyi Li, Bing Hu, and Chunxiang Wang. Restricted deformable convolution-based road scene semantic segmentation using surround view cameras. *IEEE Transactions on Intelligent Transportation Systems*, 21(10):4350–4362, 2020. 3

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5

[7] Noureldin Hendy, Cooper Sloan, Fengshi Tian, Pengfei Duan, Nick Charchut, Yuxing Xie, Chuan Wang, and James Philbin. Fishing net: Future inference of semantic heatmaps in grids. *ArXiv*, 2020. 2, 3

[8] Junjie Huang, Guan Huang, Zheng Zhu, and Dalong Du. Bevdet: High-performance multi-camera 3d object detection in bird-eye-view. *ArXiv*, 2021. 7

[9] Jason Ku, Alex D. Pon, and Steven L. Waslander. Monocular 3d object detection leveraging accurate proposals and shape reconstruction. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[10] Peixuan Li, Huaici Zhao, Pengfei Liu, and Feidao Cao. Rtm3d: Real-time monocular 3d detection from object keypoints for autonomous driving. In *ECCV*, 2020. 2

[11] Qi Li, Yue Wang, Yilun Wang, and Hang Zhao. Hdmapnet: An online hd map construction and evaluation framework. *ArXiv*, 2021. 1, 6

[12] Zongdai Liu, Dingfu Zhou, Feixiang Lu, Jin Fang, and Liangjun Zhang. Autoshape: Real-time shape-aware monocular 3d object detection. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 2

[13] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 6

[14] Yan Lu, Xinzhu Ma, Lei Yang, Tianzhu Zhang, Yating Liu, Qi Chu, Junjie Yan, and Wanli Ouyang. Geometry uncertainty projection network for monocular 3d object detection. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 2

[15] Xinzhu Ma, Wanli Ouyang, Andrea Simonelli, and Elisa Ricci. 3D Object Detection from Images for Autonomous Driving: A Survey. *arXiv e-prints*, Feb. 2022. 2

[16] F. Manhardt, W. Kehl, and A. Gaidon. Roi-10d: Monocular lifting of 2d detection to 6d pose and metric shape. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[17] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka. 3d bounding box estimation using deep learning and geometry. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[18] B. Pan, J. Sun, Hyt Leung, A. Andonian, and B. Zhou. Cross-view semantic segmentation for sensing surroundings. *IEEE Robotics and Automation Letters*, PP(99):1–1, 2020. 2, 3

[19] N. Parmar, A. Vaswani, J. Uszkoreit, Ukasz Kaiser, N. Shazeer, A. Ku, and D. Tran. Image transformer. 2018. 4

[20] Jonah Philion and Sanja Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *Proceedings of the European Conference on Computer Vision*, 2020. 1, 2, 3, 4, 5, 6, 7, 8

[21] Cody Reading, Ali Harakeh, Julia Chae, and Steven L. Waslander. Categorical depth distribution network for monocular 3d object detection. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[22] Thomas Roddick and Roberto Cipolla. Predicting semantic map representations from images using pyramid occupancy networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1, 3

[23] Thomas Roddick, Alex Kendall, and Roberto Cipolla. Orthographic feature transform for monocular 3d object detection. *British Machine Vision Conference*, 2019. 2

[24] Leslie N. Smith and Nicholay Topin. Super-convergence: very fast training of neural networks using large learning rates. In *Defense + Commercial Sensing*, 2019. 6

[25] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, 2019. 5

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv*, 2017. 3, 5

[27] Sourabh Vora, Alex H. Lang, Bassam Helou, and Oscar Beijbom. Pointpainting: Sequential fusion for 3d object detection. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1

[28] Tai Wang, Xinge Zhu, Jiangmiao Pang, and Dahua Lin. Fcos3d: Fully convolutional one-stage monocular 3d object detection. In *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2021. 2

[29] Tai Wang, Xinge ZHU, Jiangmiao Pang, and Dahua Lin. Probabilistic and geometric depth: Detecting objects in perspective. In *Proceedings of the 5th Conference on Robot Learning*, 2022. 2

[30] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Q. Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[31] Y. You, Y. Wang, W. L. Chao, D. Garg, G. Pleiss, B. Hariharan, M. Campbell, and K. Q. Weinberger. Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving. 2019. 2

[32] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end interpretable neural motion planner. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1