Make Your ViT-based Multi-view 3D Detectors Faster via Token Compression

Dingyuan Zhang[1*], Dingkang Liang[1*], Zichang Tan[2], Xiaoqing Ye[2], Cheng Zhang[1], Jingdong Wang[2], and Xiang Bai[1✉]

[1] Huazhong University of Science and Technology, Wuhan, China
{dyzhang233, dkliang, xbai}@hust.edu.cn
[2] Baidu Inc., Beijing, China

Abstract. Slow inference speed is one of the most crucial concerns for deploying multi-view 3D detectors to tasks with high real-time requirements like autonomous driving. Although many sparse query-based methods have already attempted to improve the efficiency of 3D detectors, they neglect to consider the backbone, especially when using Vision Transformers (ViT) for better performance. To tackle this problem, we explore the efficient ViT backbones for multi-view 3D detection via token compression and propose a simple yet effective method called **To**ken**C**ompression**3D** (ToC3D). By leveraging history object queries as foreground priors of high quality, modeling 3D motion information in them, and interacting them with image tokens through the attention mechanism, ToC3D can effectively determine the magnitude of information densities of image tokens and segment the salient foreground tokens. With the introduced dynamic router design, ToC3D can weigh more computing resources to important foreground tokens while compressing the information loss, leading to a more efficient ViT-based multi-view 3D detector. Extensive results on the large-scale nuScenes dataset show that our method can nearly maintain the performance of recent SOTA with up to 30% inference speedup, and the improvements are consistent after scaling up the ViT and input resolution. The code will be made at https://github.com/DYZhang09/ToC3D.

Keywords: Multi-view 3D Detection · Efficient Vision Transformer

1 Introduction

Multi-view 3D object detection is one of the most fundamental 3D vision tasks crucial for many real-world applications (*e.g.*, autonomous driving [28]), which has gained more effort and achieved great success in recent years.

Existing multi-view 3D object detection methods can be mainly categorized into two types: dense Bird's Eye View (BEV)-based [6, 7, 17] and sparse query-based [20, 36, 37]. The former extracts dense BEV features from images and

* Dingyuan Zhang and Dingkang Liang contributed equally. ✉ Corresponding author.

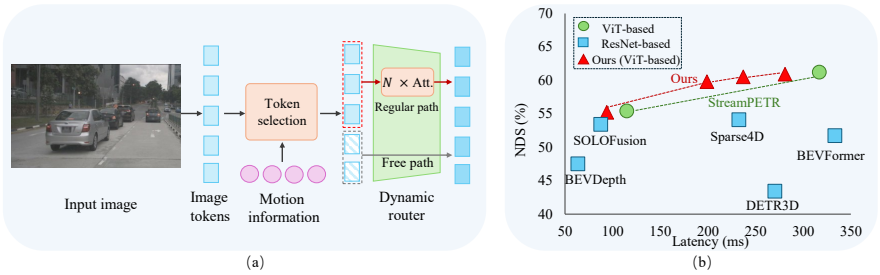


Fig. 1: (a) We trim ViTs by focusing on the foreground tokens with the aid of motion cues. (b) Our method reports an ideal trade-off between performance and latency.

then interacts with object queries to finish detection, while the latter directly uses sparse object queries to interact with image features, skipping the dense BEV feature extraction. Because sparse query-based methods mainly acquire features of 3D objects instead of whole large-scale scenes, they leverage the sparsity better, and tremendously relax the computing and memory resources requirements. However, this design puts forward higher requirements for image feature quality since sparse query-based methods cannot refine the features in BEV space. Thus, image backbones with better capacity would be beneficial.

Recently, Vision Transformers (ViTs) [3–5] have dominated the vision tasks due to their high capacity, scalability, and flexibility to integrate with multi-modal foundation models. For the sake of the performance and flexibility of 3D detection, many sparse query-based multi-view 3D detectors have been trained with advancing pre-trained ViTs. The usage of ViTs has become a trend that is increasingly prevailing. Nowadays, the sparse query-based multi-view 3D detectors [20, 23, 36] with powerful ViTs have achieved state-of-the-art performance and nearly dominated the leaderboard.

Nevertheless, although sparse query-based methods improve the efficiency by concentrating mainly on foreground objects in the 3D decoder, we experimentally found that the inference speed is not mainly hindered by the 3D decoder but by the ViT backbone. One reason is that existing sparse query-based methods use the ViTs without adjustments, treating foreground 3D objects and background things all the same. Despite the simplicity, we argue that the naive usage of ViT backbones does not obey their design principles: foreground proposals are more significant than background for 3D object detection, and we do not need to model background things in detail. This negligence brings an unnecessary burden, which motivates us to "trim" the ViT backbone to achieve better efficiency.

A simple way is to accelerate the ViT backbones for multi-view 3D detectors via token compression [29, 33, 40]. By assuming that there are only a small amount of salient foreground tokens and only these tokens need fine-grained computation, token compression methods can reweigh computation resources between foreground and background tokens. This can depress unnecessary computation and dramatically reduce the computational burden. However, existing

token compression methods are initially designed for 2D vision tasks and conduct token compression without 3D-aware features or priors. The lack of 3D awareness leads to sub-optimal token compression when facing objects with complicated 3D motion transformations. It thus significantly hurts the performance if they are applied to multi-view 3D detectors.

To accelerate the multi-view 3D detectors with ViTs while maintaining high performance, we propose a simple yet effective method called **To**ken**C**ompression **3D** (ToC3D) in this paper, shown in Fig. 1(a). The key insight is: the object queries from history predictions, which contain 3D motion information, can serve as the foreground prior of high quality. By leveraging these object queries, we can achieve 3D-aware token compression and foreground-oriented computing resource assignment. This insight allows us to further extend the philosophy of sparse query-based methods from the 3D decoder to the whole pipeline and achieve more efficient multi-view 3D object detection.

Specifically, ToC3D mainly consists of two designs: motion query-guided token selection strategy (MQTS) and dynamic router. MQTS takes image tokens and history object queries as inputs, models the motion information of object queries, and calculates the importance score of each image token through the attention mechanism. With the supervision of projected ground truth objects, it learns to divide image tokens into salient and redundant parts. Then, we pass them to the dynamic router for feature extraction of high efficiency, whose core is assigning more computing resources to the salient foreground proposals and removing unnecessary consumption for acceleration. After integrating these two modules with ViT, ToC3D further boosts the efficiency of sparse query-based multi-view 3D detectors and keeps their impressive performance.

We evaluate our method on the nuScenes [2] dataset. The extensive experiments prove the effectiveness of our method, as shown in Fig. 1(b). In detail, when compared with the StreamPETR [36] baseline, our method can nearly maintain the performance with up to 30% inference speedup, and further accelerate the baseline to the same level with other ResNet-based multi-view 3D detector [31] while keeping the performance superiority. The accuracy-efficiency tradeoff improvements are consistent after scaling up the ViT and input image resolution. Moreover, our method can also be applied to other baselines as well.

In summary, the main contributions of our method are two-fold: **1)** We point out that the naive usage of ViTs brings unnecessary computational burdens and strongly hinders the inference speed of sparse query-based multi-view 3D detectors. **2)** We propose a simple and efficient method called ToC3D to solve the problem, which uses history object queries with motion information to achieve 3D motion-aware token compression, and finally obtain faster ViTs.

2 Related Work

2.1 Multi-view 3D Object Detection

Multi-view 3D object detection has many advantages when deploying to the real world, given its low costs and simple sensor setups (*i.e.*, it only needs cameras).

Existing methods can be mainly categorized into two types: dense BEV-based paradigm [13, 16, 18, 41, 43] and sparse query-based paradigm [21, 22, 24, 25].

For dense BEV-based paradigm, many works [8, 15, 31] use the explicit view transformation (*e.g.*, LSS [32]) to transform image features into dense BEV. BEVDet [8] is the pioneer work of this paradigm. BEVDepth [15] leverages explicit depth supervision to facilitate accurate depth estimation, and SOLOFusion [31] combines long-term and short-term temporal stereo for better depth estimation, both improve the performance significantly. Instead of explicit view transformation, BEVFormer [17] pre-defines grid-shaped BEV queries and aggregates dense BEV features through attention, which is implicit. PolarFormer [9] explores the polar coordinate system to replace the grid-shaped coordinate system. Since dense BEV-based methods need to extract dense BEV features, the computation and memory costs are relatively high.

For the sparse query-based paradigm, DETR3D [37] initializes a set of 3D queries and aggregates features by projecting 3D queries into the 2D image plane. PETR [24] encodes the position information of 3D coordinates into image features, eliminating the need for 3D query projection. CAPE [39] and 3DPPE [35] further improve the quality of 3D position information. SparseBEV [23] introduces adaptability to the detector in both BEV and image space. For temporal 3D detection, Sparse4D [20] proposes sparse 4D sampling to aggregate features from multi-view/scale/timestamp. StreamPETR [36] introduces a memory queue to store history object queries for long-term temporal information propagation. Because these methods pass image features directly to the 3D decoder for detection, high-quality image features are beneficial. With advancing pre-trained ViTs, the sparse query-based methods [20, 23, 36] have achieved state-of-the-art performance and nearly dominated the leaderboard. However, their inference speeds are mainly hindered by the backbone due to the computational burden of ViT, which motivates us to trim the ViT backbone.

2.2 Token Compression for Vision Transformers

Vision transformer (ViT) [3] has gone viral in various computer vision tasks [5,38] due to its strong feature extraction ability. The visualization of trained ViT shows sparse attention maps, which means the final prediction only depends on a subset of salient tokens. Based on this observation, many works [1, 11, 26, 33, 40] attempt to speed up ViT by removing redundant tokens, dubbed token compression. Specifically, DynamicViT [33] introduces a lightweight prediction module to estimate the importance score of each token and then prunes redundant tokens progressively and dynamically. A-ViT [42] further proposes a dynamic halting mechanism. EViT [19] leverages class token attention to identify the importance of tokens, then reserves the attentive image tokens and fuses the inattentive tokens. AdaViT [29] further prunes at attention head, and block level. Si *et al.* [26] jointly considers the token importance and diversity. Evo-ViT [40] presents a self-motivated slow-fast token evolution approach, which maintains the spatial structure and information flow. All the methods are initially designed for 2D vision tasks and conduct token compression without 3D-aware priors.

In this paper, borrowing from the methods in [11,33,40], we extend the token compression from the 2D domain to the 3D domain by leveraging history object queries and modeling the 3D motion information, leading to 3D motion-aware token compression tailored for 3D object detection.

3 Method

3.1 Overview

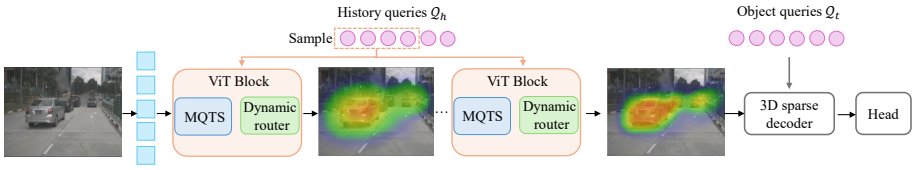
Sparse query-based methods [20,24,36] improve the efficiency of 3D detectors by mainly modeling sparse object-centric queries as the foreground proxy instead of the whole 3D scene. However, we argue that for existing sparse query-based methods, there still exists much room for efficiency improvements, as they treat foreground and background all the same in the backbone. When using ViT [4,14] to achieve extraordinary performance, the backbone becomes the bottleneck of inference speed.

To tackle the above problem, we propose to leverage token compression for extending the design philosophy of sparse query-based methods to the ViT backbone, named **To**ken**C**ompression**3D** (ToC3D). As Fig. 2(a) shows, ToC3D mainly comprises two designs: motion query-guided token selection strategy (MQTS) and dynamic router. The token compression in each block happens as follows: **1)** First, MQTS takes image tokens and history object queries as inputs and calculates the importance score of each image token through the attention between image tokens and history queries, splitting image tokens into salient and redundant ones. **2)** Then, the dynamic router is used to extract features from different groups of tokens efficiently. Salient tokens are passed to the regular path, which consists of many attention blocks. The free path with the identity layer is used for redundant tokens to save computational costs. To keep the interaction between salient and redundant tokens in attention blocks, we merge redundant tokens into one bridge token and append it with salient tokens before the regular path. **3)** Finally, after obtaining features of salient and redundant tokens, we rearrange salient and redundant tokens to meet the compatibility with typical 3D object detectors.

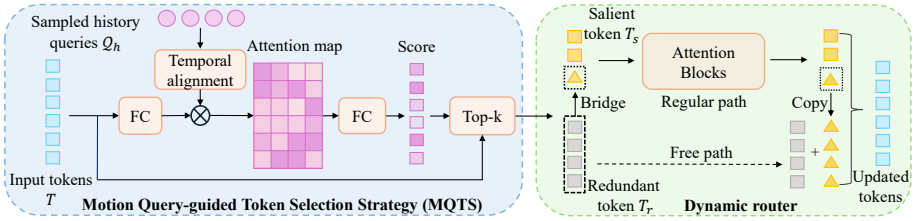
By stacking token compression-empowered blocks, the computing resources are dynamically and more intensively assigned to the foreground proposals, which removes unnecessary consumption and accelerates the inference remarkably. Ultimately, we effectively trim the ViT backbone and develop a more efficient sparse query-based multi-view 3D detector with the 3D sparse decoder.

3.2 Motion Query-guided Token Selection Strategy

Motion query-guided token selection strategy (MQTS) is meant to measure the importance score of each image token and split tokens into salient/redundant tokens. Generally, salient and redundant tokens usually contain information about foreground objects and background things. Based on this, MQTS essentially segments the foreground tokens in the image, and the history object queries with



(a) The pipeline of our ToC3D



(b) The details of our MQTS and Dynamic router

Fig. 2: (a) The overall architecture of ToC3D, which trims each block of ViT backbone through two designs: Motion Query-guided Token Selection strategy (MQTS) and dynamic router. (b) MQTS takes motion queries from history frames as inputs, calculates the importance score, and splits image tokens into salient and redundant tokens. Dynamic router passes these tokens to different paths for efficient feature extraction.

3D motion information can serve as the foreground prior of high quality, which leads to the motion query-guided token selection design shown in Fig. 2(b).

Specifically, MQTS first takes history query contents $Q_h^c \in \mathbb{R}^{N_q \times C_q}$, history query reference points $Q_h^p \in \mathbb{R}^{N_q \times 4}$ (in homogeneous coordinates), and image tokens of current frame $T \in \mathbb{R}^{N \times C}$ as inputs, where N_q, N are the number of history queries and image tokens, C_q, C are the channels of history queries and image tokens, respectively. Then, MQTS uses the following process to split tokens accurately and efficiently:

Motion query preparation. Since spatial transformation exists between current and history frames, we introduce the temporal alignment process to align Q_h^p with the current ego-coordinate system. Following [36], we use the motion-aware layer normalization. First, we view all objects as static and align Q_h^p to the current frame using the ego transformation matrix:

$$\hat{Q}^p = E_h \cdot Q_h^p, \tag{1}$$

where $\hat{Q}^p$ is the aligned history query reference points and E_h is the ego transformation matrix from history frames to current frame. Then, we model the motion of movable objects by a conditional layer normalization, with affine transformation coefficients calculated as follows:

$$V_m = \mathrm{PE}\left([v, \triangle t, E_h]\right), \quad \gamma = \mathrm{Linear}(V_m), \quad \beta = \mathrm{Linear}(V_m), \tag{2}$$

where V_m is the encoded motion vector, γ, β are affine transformation coefficients used in layer normalization, $[\cdot]$ is the concatenate operator, v is the velocity of queries, $\triangle t$ is the time difference between history queries and current frame. PE$(\cdot)$ is the positional encoding function, and we adopt the sine-cosine encoding used in NeRF [30]. Finally, we encode the motion information through conditional layer normalization:

$$\widetilde{Q}^p = \gamma \cdot \mathrm{LN}\left(\mathrm{MLP}\left(\hat{Q}^p\right)\right) + \beta, \qquad \widetilde{Q}^c = \gamma \cdot \mathrm{LN}\left(Q_h^c\right) + \beta, \tag{3}$$

where $\widetilde{Q}^c \in \mathbb{R}^{N_q \times C_q}, \widetilde{Q}^p \in \mathbb{R}^{N_q \times C_q}$ are temporally aligned history query contents and reference points embedding, and MLP$(\cdot)$ is a multi-layer perceptron used to convert reference points into embedding.

Importance score calculation. After obtaining the temporally aligned history query, we segment salient foreground tokens. To better extract the foreground prior, we leverage the attention mechanism to calculate the importance score of each image token. We first add temporally aligned history query contents and reference points embedding to get the new query embedding and then use a linear layer to align the dimensions of image tokens and query embedding:

$$\widetilde{Q} = \widetilde{Q}^p + \widetilde{Q}^c, \quad \widetilde{T} = \mathrm{Linear}\left(T\right), \tag{4}$$

where $\widetilde{Q} \in \mathbb{R}^{N_q \times C_q}, \widetilde{T} \in \mathbb{R}^{N \times C_q}$ are dimension-aligned query embedding and image tokens respectively, N, N_q are the number of image tokens and history queries, respectively. C_q is the channel number of history queries. Next, we obtain the attention map through efficient matrix multiplication:

$$A = \frac{\widetilde{T}\widetilde{Q}^\top}{\sqrt{C_q}}. \tag{5}$$

Essentially, the attention map $A \in \mathbb{R}^{N \times N_q}$ models the correlations between image tokens and history queries and thus can represent the foreground information density of each token since history queries contain foreground priors. By aggregating the foreground information density via a simple linear transformation with sigmoid activation, we determine the importance score $S \in \mathbb{R}^{N \times 1}$ of each image token:

$$S = \mathrm{Sigmoid}\left(\mathrm{Linear}\left(A\right)\right). \tag{6}$$

Finally, we select the top-k image tokens as salient tokens according to importance score, making batch processing easier:

$$T_s = \mathrm{Topk}\left(T, S, N_s\right), \quad T_r = \mathrm{Topk}\left(T, -S, N_r\right), \tag{7}$$

where $T_s \in \mathbb{R}^{N_s \times C}$ is the salient tokens and $T_r \in \mathbb{R}^{N_r \times C}$ is the redundant tokens. $N_s = \rho N, N_r = (1-\rho)\, N$ are the number of salient and redundant tokens respectively. ρ is the pre-set constant keeping ratio of each block. Increasing ρ will scale up the number of salient tokens; otherwise, scale up the number of redundant tokens, and thus, we can control the inference speed by adjusting ρ.

Although the importance score calculation is light-weight, it still brings some overhead. We found that updating the importance score of every layer does not bring noticeable improvements, so we only update the importance score before specific transformer layers for efficiency, and the layers in between will reuse the newest importance score.

History query sampling. Although history object queries carry high-quality foreground priors, we empirically found that not all history object queries are valuable. This is because the number of object queries is larger than that of objects of interest in a typical DETR-style detector, and many object queries do not correspond to foreground objects but background things. If we directly use all these history queries in MQTS, the importance score calculation will be biased by background information. Thanks to the confidence measurement of object queries, we can simply solve this problem by sampling history queries according to their confidence scores, which the decoder has already predicted in the history frames. In detail, we sort the history queries according to their confidence scores and select the top-N_q queries as inputs of MQTS.

3.3 Dynamic Router

After splitting tokens into salient and redundant ones, we introduce a dynamic router to accelerate the inference speed while keeping information losses as low as possible, shown in Fig. 2(b). Considering that not all redundant tokens correspond to the background, these tokens may contain some potential information for detection, and the information can be passed to foreground tokens via attention interactions. We merge redundant tokens into a single bridge token and append it with salient tokens, allowing interactions between salient and redundant tokens through the bridge token. Then, we use more neural layers (*i.e.*, regular path) to extract rich semantic and geometric information in salient tokens, while using shallow layers (*i.e.*, free path, we use the identity layer in this paper) to keep the information in redundant tokens.

Formally, we use the importance scores to conduct weighted sum with redundant tokens for getting bridge token:

$$T_b = \frac{\sum_{i=1}^{N_r} S_{r:i} T_{r:i}}{\sum_{i=1}^{N_r} S_{r:i}} \tag{8}$$

where $T_b \in \mathbb{R}^{1\times C}$ is the bridge token, $S_{r:i}$ is the importance score of the i-th redundant token, and $T_{r:i}$ is the i-th redundant token.

Afterwards, we append the bridge token after salient tokens and pass them to the regular path for extracting rich semantic and geometric features:

$$[T'_s, T'_b] = \text{Blocks}\left([T_s, T_b]\right), \tag{9}$$

where T'_s, T'_b are updated salient and bridge tokens, Blocks are transformer encoder blocks, typically consisting of several window attention layers and a global attention layer for multi-view 3D object detection.

For redundant tokens, we pass them to the fast path (*i.e.*, identity layer) and add them with the updated bridge token:

$$T_r' = T_r + \text{Repeat}(T_b', N_r), \tag{10}$$

where T_r' is the updated redundant tokens and $\text{Repeat}(x, y)$ repeats x by y times.

Finally, we combine the updated salient and redundant tokens to obtain the updated image tokens. Thanks to this simple yet effective dynamic router, we refine image tokens more efficiently and meet the compatibility with typical multi-view 3D detectors.

4 Experiments

4.1 Dataset and Metrics

We evaluate our method on the large-scale nuScenes [2] dataset, consisting of 700 scenes for training, 150 for validation, and 150 for testing. The data of each scene is captured by six cameras at 10Hz, with full 360° field of view (FOV). We use annotations of 10 classes: car, truck, construction vehicle, bus, trailer, barrier, motorcycle, bicycle, pedestrian, and traffic cone. We use the official nuScenes metrics for comparison: the nuScenes detection score (NDS), the mean average precision (mAP), the average translation error (ATE), the average scale error (ASE), the average orientation error (AOE), the average velocity error (AVE), average attribute error (AAE).

4.2 Implementation Details

We select recently representative StreamPETR [36] as our basic pipeline, considering its high performance. For the backbone, we adopt ViT-B, ViT-L [3] and conduct token compression on them. We use the Gaussian Focal Loss [12] to supervise the MQTS, with the ground truth coming from projected bounding boxes. The model is trained on 8 NVIDIA V100 with a total batch size of 16 for 24 epochs. The inference speed is tested on a single RTX3090. AdamW [27] is used as the optimizer. The augmentation follow the StreamPETR [36], and without CBGS [44]. The detailed configurations can be viewed in Tab. 1.

4.3 Main Results

We compare our method with the basic pipeline StreamPETR [36] and other popular multi-view 3D detectors on nuScenes [2] val set.

The main results are illustrated in Tab. 2. When using ViT-B as the backbone, our method (ToC3D-Fast) can perfectly maintain NDS and mAP compared with the StreamPETR method, with nearly 20% speedup. If marginal 0.5% NDS and mAP drop are allowed, our method (ToC3D-Faster) can further accelerate the backbone by 30% and the whole pipeline by 26%. Notably,

Table 1: Details of settings.

Configurations	ToC3D-Fast		ToC3D-Faster	
Backbone	ViT-B	ViT-L	ViT-B	ViT-L
Dim. of image token C	768	1024	768	1024
Num. of layers	12	24	12	24
Num. of object query N_q	64	64	64	64
Dim. of object query C_q	256	256	256	256
Keeping ratios ρ	0.7, 0.5, 0.5	0.7, 0.5, 0.5	0.5, 0.4, 0.3	0.5, 0.4, 0.3
Loc. of importance score S updating	3, 6, 9	6, 12, 18	3, 6, 9	6, 12, 18
Token compression loss weight	5.0	5.0	5.0	5.0
Pretrained weight	SAM [10]	EVA-02 [4]	SAM [10]	EVA-02 [4]

Table 2: The main results on the nuScenes val set. We report the backbone inference time (before the slash) and the whole pipeline inference time (after the slash) to illustrate the impact of efficient backbone better. †means using larger image resolution.

Method	Backbone	NDS(%)↑	mAP(%)↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓	Infer. Time (ms)↓
BEVDet [8]	R50	37.9	29.8	0.725	0.279	0.589	0.860	0.245	- / 59.9
BEVDepth [15]	R50	47.5	35.1	0.639	0.267	0.479	0.428	0.198	- / 63.7
SOLOFusion [31]	R50	53.4	42.7	0.567	0.274	0.511	0.252	0.181	- / 87.7
StreamPETR [36]	ViT-B	55.4	45.8	0.608	0.272	0.415	0.261	0.191	85.2 / 115.0
Ours-Fast	ViT-B	55.2	46.0	0.604	0.270	0.449	0.261	0.196	70.3(-17.5%) / 94.0(-18.3%)
Ours-Faster	ViT-B	54.9	45.3	0.594	0.271	0.443	0.258	0.207	59.2(-30.5%) / 85.0(-26.1%)
DETR3D† [37]	R101	43.4	34.9	0.716	0.268	0.379	0.842	0.200	- / 270.3
BEVFormer† [17]	R101	51.7	41.6	0.673	0.274	0.372	0.394	0.198	- / 333.3
Sparse4D† [20]	R101	54.1	43.6	0.633	0.279	0.363	0.317	0.177	- / 232.6
StreamPETR [36]	ViT-L	61.2	52.1	0.552	0.251	0.249	0.237	0.196	290.0 / 317.0
Ours-Fast	ViT-L	60.9	51.7	0.552	0.250	0.268	0.229	0.195	253.0(-12.8%) / 281.0(-11.4%)
Ours-Faster	ViT-L	60.5	51.3	0.562	0.250	0.265	0.230	0.203	209.0(-28.0%) / 237.2(-25.2%)
StreamPETR† [36]	ViT-L	62.7	55.8	0.552	0.256	0.287	0.225	0.201	1222.4 / 1309.9
Ours-Fast†	ViT-L	62.6	54.9	0.536	0.254	0.259	0.230	0.206	964.8(-21.1%) / 1051.9(-19.7%)
Ours-Faster†	ViT-L	61.9	54.3	0.560	0.257	0.230	0.234	0.201	791.0(-35.3%) / 878.5(-33.0%)

ToC3D-Faster performs at the same level as StreamPETR while only costs like SOLOFusion with R50 backbone, indicating the effectiveness of our method.

When using ViT-L as the backbone, our method (ToC3D-Fast) achieves nearly lossless performance compared to the basic StreamPETR while accelerating the whole pipeline by 36ms. Furthermore, with a performance loss of no more than 0.9%, our method (ToC3D-Faster) brings 25% inference speed gains and runs at the same speed with Sparse4D [20] while keeping the vast performance superiority (*i.e.*, over 6.4% NDS and 7.7% mAP).

Furthermore, scaling up the input image resolution to 800×1600, our method can tremendously reduce the inference time by 258ms and 431ms with ToC3D-Fast and ToC3D-Faster settings, saving a considerable amount of computing resources for detection deployed on the cloud. The results prove that our method can achieve better trade-offs and greatly improve the efficiency of 3D detectors.

4.4 Analysis

We conduct experiments for analysis of our method using ViT-L as the backbone. All models are trained for only 12 epochs and evaluated on the val set.

Compared to 2D token compression methods. To prove the effectiveness of our motion query-guided token selection strategy (MQTS), we compare

Table 3: Results of comparison between our method with different 2D token compression methods on the nuScenes val set. We apply these methods to the StreamPETR baseline and have carefully tuned their hyper-parameters to achieve their best results.

Method	Type	Keeping Ratio ρ	NDS(%)↑	mAP(%)↑	Infer. Time (ms)↓
StreamPETR [36]	-	-	61.2	52.1	290.0 / 317.0
+ Random	Random	0.7, 0.5, 0.5	56.7	46.5	250.1 / 277.9
+ Random	Random	0.5, 0.4, 0.3	48.5	36.0	207.3 / 235.1
+ DynamicViT [33]	Score-based	0.7, 0.5, 0.5	59.7	50.5	249.8 / 277.4
+ DynamicViT [33]	Score-based	0.5, 0.4, 0.3	59.3	49.3	208.0 / 233.4
+ SparseDETR [34]	Score-based	0.7, 0.5, 0.5	59.3	49.7	249.0 / 280.5
+ SparseDETR [34]	Score-based	0.5, 0.4, 0.3	59.1	49.2	208.5 / 236.5
Ours-Fast	Motion Query-guided	0.7, 0.5, 0.5	61.0	52.3	253.0 / 281.0
Ours-Faster	Motion Query-guided	0.5, 0.4, 0.3	60.3	51.2	209.0 / 237.2

our method with typical 2D token compression methods DynamicViT [33] and SparseDETR [34]. For a fair comparison, we replace the MQTS with these two methods, keep the dynamic route unchanged, and tune these methods to their best performance. We also compare with the Random token compression.

As listed in Tab. 3, it clearly shows that the random token compression brings a significant performance drop, especially when keeping ratios are low. This is because the random compression cannot capture the importance of image tokens and thus drops much helpful information. When using the score-based 2D token compression methods, the performance drop is much smaller than random compression since they are better aware of important foreground tokens and thus suffer less information loss. However, because these 2D methods only take image tokens as input, they conduct token compression without any 3D-aware features or priors. The lack of 3D awareness leads to sub-optimal token compression and thus hurts the performance severely (*i.e.*, about 2% mAP and 2% NDS).

When it comes to our method, because MQTS has history object queries as inputs, it can model the 3D motion information of objects and aggregate the rich 3D foreground priors of high quality, leading to remarkably better results than 2D competitors (more than 2% mAP and 1.2% NDS improvement). Notably, With efficient MQTS, our method is able to almost maintain the performance of the basic pipeline at the same speed level as 2D token compression methods, indicating the superiority of MQTS.

Effectiveness of components. After proving our key insight that the object queries from history predictions can serve as the foreground prior of high quality, we now study what makes this insight work. We take our method with the Faster setting as the baseline of this experiment, and we remove one component each time to measure its effectiveness, shown in Tab. 4. It is worth noting that removing any components only slightly reduces inference time, showing the high efficiency of each component.

For setting (a), we replace the attention in Eq. 5 with a lightweight module, which brings 0.8% mAP and 0.4% NDS drop. This is because the attention map naturally models the correlations between image tokens and history queries and

Table 4: Effectiveness of different components on the nuScenes val set. **Attn.** means calculating importance score through the attention mechanism. **Motion** means using motion vector encoding. **Samp. Q.** means using the sampled history queries as the inputs of MQTS. **Bri. T.** means using the bridge token in the dynamic router.

Setting	Attn.	Motion	Samp. Q.	Bri. T.	NDS(%)↑	mAP(%)↑	Infer. Time (ms)↓
Ours	✓	✓	✓	✓	60.3	51.2	209.0 / 237.2
(a)		✓	✓	✓	59.9	50.4	206.9 / 235.2
(b)	✓		✓	✓	59.7	50.1	206.4 / 234.5
(c)	✓	✓		✓	59.9	50.4	209.3 / 237.3
(d)	✓	✓	✓		60.2	50.5	203.1 / 232.7

Table 5: Effect of N_q on the nuScenes val set.

N_q	NDS(%)	mAP(%)
16	60.1	50.8
32	60.3	51.0
64	**60.3**	**51.2**
128	60.1	50.6
256	59.9	50.4

Table 6: Effect of keeping ratios on the nuScenes val set.

ρ	NDS(%)	mAP(%)	Infer. Time (ms)
StreamPETR [36]	61.2	52.1	290.0 / 317.0
0.7, 0.5, 0.5	61.0	52.3	253.0(-12.8%) / 281.0(-11.4%)
0.7, 0.5, 0.3	60.7	51.6	235.9(-18.7%) / 264.6(-16.5%)
0.5, 0.4, 0.3	60.3	51.2	209.0(-28.0%) / 237.2(-25.2%)
0.4, 0.3, 0.2	59.9	50.5	185.8(-36.0%) / 217.0(-31.5%)
0.4, 0.3, 0.1	59.8	50.4	172.1(-40.7%) / 199.0(-37.2%)
0.3, 0.2, 0.1	59.0	49.1	155.8(-46.3%) / 183.3(-42.2%)

thus more explicitly represents the foreground information density of each token, leading to better importance measurement.

For setting (b), we discard processes from Eq. 1∼3. The degraded performance (1.1% mAP and 0.6% NDS) shows the importance of motion information, which adaptively handles movable objects and suppresses the noise brought by misalignment between history objects and the current coordinate system.

For setting (c), we remove the history query sampling and use all history queries instead. This choice reduces mAP by 0.8% and NDS by 0.4%, showing the necessity of history query sampling, as it removes object queries corresponding to background things and prevents importance score calculation from being biased.

For setting (d), we do not use the bridge token in the dynamic router. Because of the absence of interaction between salient and redundant tokens, potential information contained in redundant tokens can not be passed to foreground tokens, leading to information loss and is ultimately reflected in mAP and NDS.

Impact of history query num N_q. Since we sample N_q history queries as inputs of MQTS, we study the impact of different N_q in this section, shown in Tab. 5. When $N_q = 256$, we use all history queries. Otherwise, we sample top-N_q history queries according to their confidence scores from history predictions. The results show that sampling history queries is always beneficial, as many history queries correspond to background things. History query sampling helps prevent the importance score from being biased by background and thus improves the performance. However, if N_q is too small, it will drop many foreground queries, suffer information loss, and hurt the performance. We empirically find that $N_q = 64$ can achieve better performance.

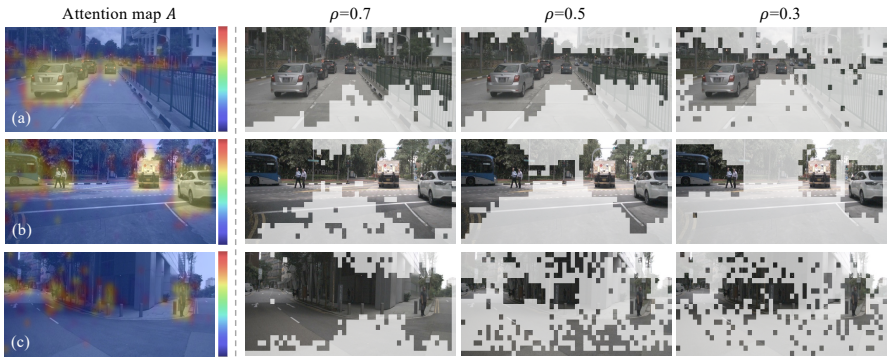


Fig. 3: The visualization of our method (better viewed in color). We visualize the attention map in importance score calculation on the left and the salient/redundant tokens after the top-k selection on the right. Redundant tokens are illustrated as translucent.

Table 7: Analysis of generalization on the nuScenes val set. We report the backbone inference time before the slash and the whole pipeline inference time after the slash.

Method	Backbone	Compression	NDS(%)	mAP(%)	Infer. Time (ms)
StreamPETR [36]	ViT-L	-	61.2	52.1	290.0 / 317.0
Ours-Faster	ViT-L	✓	60.3	51.2	209.0(-28.0%) / 237.2(-25.2%)
Sparse4Dv2 [21]	ViT-L	-	58.8	50.9	278.8 / 322.0
Ours-Faster	ViT-L	✓	58.8	50.1	206.6(-25.9%) / 244.8(-24.0%)

Impact of keeping ratios ρ. Keeping ratios ρ controls the number of salient tokens and decides the accuracy-speed trade-off. In this section, we conduct experiments to figure out how keeping ratios affects the efficiency of our method. From Tab. 6, we can get the following phenomenon: (1) We can speed up the pipeline by nearly 15% without noticeable performance loss, and 25% with marginal NDS loss (0.9%). (2) In a certain range, the performance drop is nearly linear with the inference time drop (about 0.2% NDS drop for 20ms inference time). (3) Too small keeping ratios (*i.e.*, 0.3, 0.2, 0.1) will bring a significant performance drop. This is because foreground tokens usually account for more than 10% image tokens, and too small keeping ratios inevitably discard foreground tokens, bringing information loss. Considering the requirements of real applications, we set the model with keeping ratios of 0.7, 0.5, 0.5 as ToC3D-Fast, and 0.5, 0.4, 0.3 as ToC3D-Faster versions, respectively, as these models have relatively better trade-off.

Generalization. We select StreamPETR [36] as our basic pipeline, but this does NOT mean that the application of our method is limited. In fact, ToC3D can serve as a plug-and-play method, and we show the generalization ability of our method by applying it to another strong pipeline, Sparse4Dv2 [21]. The results are shown in Tab. 7. It indicates that the behavior of our method is consistent across different baseline methods. With the same speed-up ratio as

on StreamPETR, our approach keeps the performance loss within 0.8% and surprisingly maintains the exact NDS compared to the strong Sparse4Dv2 [21], proving the feasibility and effectiveness of our method on other pipelines.

4.5 Qualitative Results

To better study the behavior of MQTS, we visualize the attention map and salient tokens in Fig. 3. The attention maps clearly show that our method focuses on foreground objects of interest precisely, no matter the large objects (*e.g.*, cars and trucks in sample (a) or the small objects (*e.g.*, pedestrians in sample (b) and (c)). This is an intuitive proof of our claim in Sec. 3.2, *i.e.*, the attention map models the correlations between image tokens and history queries, and thus can represent the foreground information density of each token since history queries contain foreground priors. With the 3D foreground object-aware attention map, the whole model can be more concentrated on foreground tokens when keeping ratios ρ getting lower, improving the efficiency.

4.6 Limitations

Since our method leverages history object queries as high-quality foreground priors, we assume the inputs are temporal image sequences with contiguous information. Although this assumption narrows the application of our method, we argue that this assumption is not strong as the perception system runs at typical frequencies and perceives environments contiguously for real-world autonomous driving. The second limitation of our method is that we need to re-train the token compression model if keeping ratios are changed. Using dynamic keeping ratios with some technical tricks for stability when training would help to partially solve this limitation, which is left for our future work.

5 Conclusion

In this paper, we claim that the naive usage of ViTs brings unnecessary computational burden and strongly hinders the speed of existing sparse query-based multi-view 3D detectors. To obtain a more efficient sparse multi-view 3D detector, we propose a simple yet effective method called ToC3D. Equipped with MQTS and dynamic router, ToC3D leverages history object queries as foreground priors of high quality, models 3D motion information in them, and weighs more computing resources to important foreground tokens while compressing the information loss. By doing so, we extend the design philosophy of sparse query-based methods from the 3D decoder to the whole pipeline. The experiments on the large-scale nuScenes dataset show that our method can boost the inference speed with marginal performance loss, and using history object queries brings better results. We hope this paper can inspire the research of efficient multi-view 3D detectors and serve as a strong baseline.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (Grant. No. 62225603 and 623B2038), and the Hubei Key R&D Program (Grant No. 2022BAA078).

References

1. Bolya, D., Fu, C.Y., Dai, X., Zhang, P., Feichtenhofer, C., Hoffman, J.: Token merging: Your vit but faster. ICLR (2023)
2. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. In: CVPR. pp. 11621–11631 (2020)
3. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. ICLR (2020)
4. Fang, Y., Sun, Q., Wang, X., Huang, T., Wang, X., Cao, Y.: Eva-02: A visual representation for neon genesis. arXiv preprint arXiv:2303.11331 (2023)
5. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. In: CVPR. pp. 16000–16009 (2022)
6. Huang, J., Huang, G.: Bevdet4d: Exploit temporal cues in multi-camera 3d object detection. arXiv preprint arXiv:2203.17054 (2022)
7. Huang, J., Huang, G.: Bevpoolv2: A cutting-edge implementation of bevdet toward deployment. arXiv preprint arXiv:2211.17111 (2022)
8. Huang, J., Huang, G., Zhu, Z., Ye, Y., Du, D.: Bevdet: High-performance multi-camera 3d object detection in bird-eye-view. arXiv preprint arXiv:2112.11790 (2021)
9. Jiang, Y., Zhang, L., Miao, Z., Zhu, X., Gao, J., Hu, W., Jiang, Y.G.: Polarformer: Multi-camera 3d object detection with polar transformer. In: AAAI. vol. 37, pp. 1042–1050 (2023)
10. Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.Y., et al.: Segment anything. ICCV (2023)
11. Kong, Z., Dong, P., Ma, X., Meng, X., Niu, W., Sun, M., Shen, X., Yuan, G., Ren, B., Tang, H., et al.: Spvit: Enabling faster vision transformers via latency-aware soft token pruning. In: ECCV. pp. 620–640. Springer (2022)
12. Law, H., Deng, J.: Cornernet: Detecting objects as paired keypoints. In: ECCV. pp. 734–750 (2018)
13. Li, Y., Huang, B., Chen, Z., Cui, Y., Liang, F., Shen, M., Liu, F., Xie, E., Sheng, L., Ouyang, W., et al.: Fast-bev: A fast and strong bird's-eye view perception baseline. IEEE Transactions on Pattern Analysis and Machine Intelligence (2024)
14. Li, Y., Mao, H., Girshick, R., He, K.: Exploring plain vision transformer backbones for object detection. In: ECCV. pp. 280–296. Springer (2022)
15. Li, Y., Ge, Z., Yu, G., Yang, J., Wang, Z., Shi, Y., Sun, J., Li, Z.: Bevdepth: Acquisition of reliable depth for multi-view 3d object detection. In: AAAI. vol. 37, pp. 1477–1485 (2023)
16. Li, Z., Lan, S., Alvarez, J.M., Wu, Z.: Bevnext: Reviving dense bev frameworks for 3d object detection. CVPR (2024)
17. Li, Z., Wang, W., Li, H., Xie, E., Sima, C., Lu, T., Qiao, Y., Dai, J.: Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In: ECCV. pp. 1–18. Springer (2022)

18. Li, Z., Yu, Z., Wang, W., Anandkumar, A., Lu, T., Alvarez, J.M.: Fb-bev: Bev representation from forward-backward view transformations. In: ICCV. pp. 6919–6928 (2023)
19. Liang, Y., Ge, C., Tong, Z., Song, Y., Wang, J., Xie, P.: Not all patches are what you need: Expediting vision transformers via token reorganizations. ICLR (2022)
20. Lin, X., Lin, T., Pei, Z., Huang, L., Su, Z.: Sparse4d: Multi-view 3d object detection with sparse spatial-temporal fusion. arXiv preprint arXiv:2211.10581 (2022)
21. Lin, X., Lin, T., Pei, Z., Huang, L., Su, Z.: Sparse4d v2: Recurrent temporal fusion with sparse model. arXiv preprint arXiv:2305.14018 (2023)
22. Lin, X., Pei, Z., Lin, T., Huang, L., Su, Z.: Sparse4d v3: Advancing end-to-end 3d detection and tracking. arXiv preprint arXiv:2311.11722 (2023)
23. Liu, H., Teng, Y., Lu, T., Wang, H., Wang, L.: Sparsebev: High-performance sparse 3d object detection from multi-camera videos. In: ICCV. pp. 18580–18590 (2023)
24. Liu, Y., Wang, T., Zhang, X., Sun, J.: Petr: Position embedding transformation for multi-view 3d object detection. In: ECCV. pp. 531–548. Springer (2022)
25. Liu, Y., Yan, J., Jia, F., Li, S., Gao, A., Wang, T., Zhang, X.: Petrv2: A unified framework for 3d perception from multi-camera images. In: ICCV. pp. 3262–3272 (2023)
26. Long, S., Zhao, Z., Pi, J., Wang, S., Wang, J.: Beyond attentive tokens: Incorporating token importance and diversity for efficient vision transformers. In: CVPR. pp. 10334–10343 (2023)
27. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: ICLR (2018)
28. Ma, X., Ouyang, W., Simonelli, A., Ricci, E.: 3d object detection from images for autonomous driving: a survey. IEEE Transactions on Pattern Analysis and Machine Intelligence (2023)
29. Meng, L., Li, H., Chen, B.C., Lan, S., Wu, Z., Jiang, Y.G., Lim, S.N.: Adavit: Adaptive vision transformers for efficient image recognition. In: CVPR. pp. 12309–12318 (2022)
30. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021)
31. Park, J., Xu, C., Yang, S., Keutzer, K., Kitani, K., Tomizuka, M., Zhan, W.: Time will tell: New outlooks and a baseline for temporal multi-view 3d object detection. ICLR (2022)
32. Philion, J., Fidler, S.: Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In: ECCV. pp. 194–210. Springer (2020)
33. Rao, Y., Zhao, W., Liu, B., Lu, J., Zhou, J., Hsieh, C.J.: Dynamicvit: Efficient vision transformers with dynamic token sparsification. NeurIPS **34**, 13937–13949 (2021)
34. Roh, B., Shin, J., Shin, W., Kim, S.: Sparse detr: Efficient end-to-end object detection with learnable sparsity. In: ICLR (2021)
35. Shu, C., Deng, J., Yu, F., Liu, Y.: 3dppe: 3d point positional encoding for transformer-based multi-camera 3d object detection. In: ICCV. pp. 3580–3589 (2023)
36. Wang, S., Liu, Y., Wang, T., Li, Y., Zhang, X.: Exploring object-centric temporal modeling for efficient multi-view 3d object detection. ICCV (2023)
37. Wang, Y., Guizilini, V.C., Zhang, T., Wang, Y., Zhao, H., Solomon, J.: Detr3d: 3d object detection from multi-view images via 3d-to-2d queries. In: Conference on Robot Learning. pp. 180–191. PMLR (2022)

38. Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J.M., Luo, P.: Segformer: Simple and efficient design for semantic segmentation with transformers **34**, 12077–12090 (2021)
39. Xiong, K., Gong, S., Ye, X., Tan, X., Wan, J., Ding, E., Wang, J., Bai, X.: Cape: Camera view position embedding for multi-view 3d object detection. In: CVPR. pp. 21570–21579 (2023)
40. Xu, Y., Zhang, Z., Zhang, M., Sheng, K., Li, K., Dong, W., Zhang, L., Xu, C., Sun, X.: Evo-vit: Slow-fast token evolution for dynamic vision transformer. In: AAAI. vol. 36, pp. 2964–2972 (2022)
41. Yang, C., Chen, Y., Tian, H., Tao, C., Zhu, X., Zhang, Z., Huang, G., Li, H., Qiao, Y., Lu, L., et al.: Bevformer v2: Adapting modern image backbones to bird's-eye-view recognition via perspective supervision. In: CVPR. pp. 17830–17839 (2023)
42. Yin, H., Vahdat, A., Alvarez, J.M., Mallya, A., Kautz, J., Molchanov, P.: A-vit: Adaptive tokens for efficient vision transformer. In: CVPR. pp. 10809–10818 (2022)
43. Zhang, D., Liang, D., Yang, H., Zou, Z., Ye, X., Liu, Z., Bai, X.: Sam3d: Zero-shot 3d object detection via segment anything model. Science China Information Sciences (2024)
44. Zhu, B., Jiang, Z., Zhou, X., Li, Z., Yu, G.: Class-balanced grouping and sampling for point cloud 3d object detection. arXiv preprint arXiv:1908.09492 (2019)