PQR3D: PROGRESSIVE QUERY REFINEMENT OVER REFERENCE-CONDITIONED TEMPORAL WINDOWS FOR MULTI-VIEW 3D OBJECT DETECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Temporal context is essential for camera-only multi-view 3D object detection. Existing streaming detectors maintain and propagate query states from one frame to the next, requiring sequence-aware training and chronological inference. We propose PQR3D, which performs progressive query refinement within reference-conditioned temporal windows. This design enables random frame sampling and independent inference without persistent query memory. Within each window, PQR3D progressively transfers motion-aligned high-confidence queries from earlier timestamps toward the target frame. We further introduce masked self-attention to regulate interactions among regular, propagated, and denoising queries while keeping denoising supervision isolated from detection queries. In addition, a stage-decoupled anchor embedding injects position before self-attention and size, orientation, and velocity afterward, reducing interference from temporally inconsistent attributes. Extensive experiments on nuScenes demonstrate consistent improvements across multiple backbones. With a ViT-L backbone, PQR3D sets a new state of the art on the nuScenes test set, achieving 71.6 NDS and 64.9 mAP. Source code is available at `https://anonymous.4open.science/r/pqr3d-review-B5E8`

1 INTRODUCTION

3D perception is a fundamental task in autonomous vehicles, robotics, virtual reality, and augmented reality. Multiple sensors, including cameras, LiDAR, and radar, can be leveraged to improve perception performance. Among them, cameras capture rich semantic cues, such as object boundaries, textures, and background context, which closely align with the human visual system.

Recent query-based camera-only methods use 3D reference points to aggregate image features for 3D object detection. With temporal modeling, they achieve strong accuracy and efficiency. Existing temporal query-based methods mainly follow two strategies. The first directly samples and aggregates image features from multiple timestamps for each current-frame query Lin et al. (2022); Liu et al. (2023a). However, it fuses past features directly rather than progressively refining queries across frames. The second sequentially propagates cached query states between video frames. StreamPETR Wang et al. (2023a) stores high-confidence object queries in a memory queue and propagates them frame by frame, while Sparse4Dv2 and Sparse4Dv3 Lin et al. (2023a;b) transfer instance features and structured anchors from the preceding frame to the current frame. These methods effectively model long-term temporal information but require chronological processing because each prediction depends on states from earlier frames.

However, sequential query propagation with persistent memory introduces several limitations. First, the model must preserve scene boundaries and chronological order during training and inference, preventing target frames from being processed as fully independent samples. This requirement limits random target-frame sampling during training and prevents frames from being processed independently during inference. Second, inaccurate queries from earlier frames can be propagated forward and affect predictions in later frames. Finally, these methods must maintain a temporal memory and reset it whenever the input video sequence changes. As illustrated in Figure 1, both StreamPETR and Sparse4D v2 maintain persistent states across successive frames.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

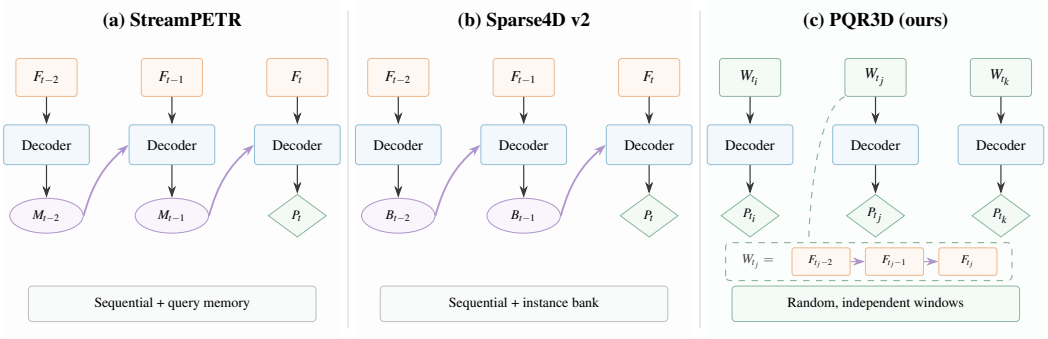


Figure 1: **Comparison of temporal propagation strategies.** (a) StreamPETR processes frames sequentially and passes query memory M_t from one frame to the next. (b) Sparse4D v2 performs sequential propagation by passing the instance bank B_t from one frame to the next. (c) PQR3D processes each reference-conditioned temporal window independently, without carrying memory across target samples. The dashed box shows one window, $W_{t_j} = \{F_{t_j-2}, F_{t_j-1}, F_{t_j}\}$, whose features are processed in temporal order to produce the 3D detection result P_{t_j} for the target frame. Here, F_t and P_t denote the multi-view image features and 3D detection result at time t, respectively.

To address these limitations, we propose PQR3D, a framework for progressive query refinement over reference-conditioned temporal windows. As shown in Figure 1, PQR3D constructs a self-contained window, such as $W_{t_j} = \{F_{t_j-2}, F_{t_j-1}, F_{t_j}\}$, for each target frame. Within each window, object queries are progressively refined from earlier timestamps toward the target timestamp. At each step, temporal features are aligned to the current frame, and high-confidence queries are motion-compensated before being passed to the next step. After predicting the target frame, PQR3D discards the local state, allowing each temporal window to be processed independently.

In addition, to combine query denoising with propagated queries within each temporal window, we introduce masked self-attention that isolates denoising queries from regular and propagated queries while allowing full interaction between the latter two groups. Furthermore, Sparse4D v3 separately encodes multiple anchor attributes, including position, size, orientation, and velocity, and incorporates the resulting representation into attention Lin et al. (2023b). However, directly injecting the complete anchor embedding before self-attention degrades detection performance in PQR3D, where regular and propagated queries interact within the same temporal window. We therefore propose a stage-decoupled anchor embedding (SDAE), which injects different anchor attributes at different stages of the decoder. The position embedding is added before self-attention to guide spatial interactions, whereas the size, orientation, and velocity embeddings are injected afterward to refine object geometry and motion.

Our main contributions are summarized as follows: 1) We present PQR3D, a progressive query refinement framework that processes each reference-conditioned temporal window independently and refines object queries in temporal order toward the target frame, without carrying memory across target samples. 2) We introduce a masked self-attention mechanism tailored to progressive query refinement, which isolates denoising queries from regular and propagated queries while preserving temporal interactions between the latter two. 3) We propose a stage-decoupled anchor embedding that injects position before self-attention and size, orientation, and velocity afterward, separating spatial query interaction from geometric and motion refinement. 4) PQR3D achieves state-of-the-art performance on the nuScenes benchmark. It consistently outperforms existing methods across different backbones, while requiring fewer training epochs. The code is available in an anonymous repository at `https://anonymous.4open.science/r/pqr3d-review-B5E8`

2 Related Work

2.1 BEV-based Methods

BEV-based methods explicitly predict dense depth to lift 2D image features into 3D space, which are then projected onto the BEV plane to create BEV grid features. BEVDet Huang et al. (2021) builds upon the LSS Philion & Fidler (2020) paradigm and introduces a BEV-based 3D detection

framework with specifically designed data augmentation and NMS strategies. BEVDepth Li et al. (2022b) enhances the LSS paradigm with explicit depth supervision from LiDAR point clouds and camera-aware depth prediction, improving depth estimation quality. AeDet Feng et al. (2022) proposes an azimuth-equivariant convolution (AeConv) and an azimuth-equivariant anchor to preserve the inherent radial symmetry of BEV features and facilitate detector optimization.

With advanced temporal modeling that leverages multi-frame BEV features, recent BEV-based methods have achieved superior performance over single-frame methods. BEVDet4D Huang & Huang (2022a) extends BEVDet with temporal fusion by aligning the previous frame's BEV feature to the current frame via ego-motion compensation and concatenating the two. BEVStereo Li et al. (2023a) improves depth estimation by introducing temporal multi-view stereo that utilizes adjacent frames for stereo matching, with dynamic candidate selection to reduce computational cost. BEVFormer series Li et al. (2022c); Yang et al. (2023) propose unified BEV representations learned via spatiotemporal transformers, using spatial cross-attention to aggregate multi-view features into predefined BEV queries and temporal self-attention to recurrently fuse historical BEV features. SOLOFusion Park et al. (2023) combines long-term, coarse stereo matching with short-term, fine-grained matching, showing that the two are complementary for camera-based 3D detection. HoP Zong et al. (2023) introduces an auxiliary task that predicts objects at a historical timestamp $t-k$ from a pseudo BEV feature, improving spatial and temporal representations without inference overhead.

2.2 Sparse query-based methods

Query-based methods initialize 3D reference points within the detection range and use them to aggregate image features. DETR3D Wang et al. (2022) uses sparse 3D object queries to sample multi-view 2D features through camera transformations. The PETR series Liu et al. (2022; 2023b) encodes 3D positional information into image features, allowing object queries to interact directly with position-aware representations for end-to-end detection.

Recent query-based methods have substantially improved detection performance by incorporating temporal modeling over multi-frame image features. The Sparse4D series Lin et al. (2022; 2023a;b) progressively improves sparse query-based detection. Sparse4D samples multi-view, multi-scale, multi-frame features through projected 3D keypoints. Sparse4Dv2 propagates sparse features recurrently for long-term modeling at reduced cost. Sparse4Dv3 adds temporal instance denoising, quality estimation, and decoupled attention. StreamPETR Wang et al. (2023a) proposes an object-centric temporal mechanism that propagates historical information through object queries frame by frame, along with a motion-aware layer normalization to capture object motion. Building upon StreamPETR, RayDN Liu et al. (2024) introduces ray denoising, which samples hard negatives along camera rays based on depth distributions, encouraging the model to learn depth-aware features for more accurate detection. SparseBEV Liu et al. (2023a) introduces query-guided adaptive spatio-temporal sampling and adaptive mixing, which decode sampled features through dynamic weights derived from the queries. Far3D Jiang et al. (2024) leverages high-quality 2D object priors to generate 3D adaptive queries that complement the 3D global queries, and introduces a perspective-aware aggregation module to effectively aggregate multi-view, multi-scale features for long-range detection. The HENet series Xia et al. (2024; 2025) introduces a hybrid image encoding strategy, applying a lightweight encoder to long-term frames and a large encoder to short-term frames, with an attention-based module to integrate temporal features from both encoders. In contrast to direct multi-frame feature aggregation and persistent memory sequential propagation, PQR3D performs progressive query refinement within reference-conditioned temporal windows, each constructed and processed independently. Within each window, PQR3D aligns temporal features and progressively refines queries toward the target frame. The local state is discarded after prediction, enabling random target-frame sampling during training and independent inference without persistent memory.

3 Method

As shown in Fig. 2, PQR3D follows an encoder-decoder architecture with progressive query refinement. Given multi-view multi-frame images at timestamp t, an image encoder (e.g., ResNet-50) with FPN extracts the feature maps to form the multi-frame feature sequence $\mathcal{F}_t = \{\mathbf{F}_t, \mathbf{F}_{t-1}, \cdots, \mathbf{F}_{t-N}\}$, where $\mathbf{F}_t$ denotes the multi-view multi-scale feature at timestamp t. A set of learnable object queries $\mathbf{Q}_t$ are initialized in the detection range of BEV space, where each query

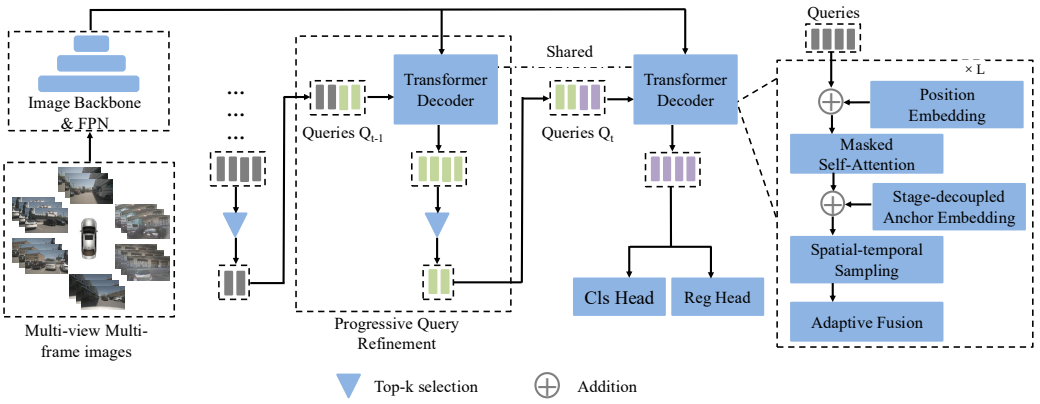


Figure 2: **Overall architecture of PQR3D.** Within each temporal window, high-confidence queries are propagated from earlier frames toward the current frame, concatenated with the current queries Q_t, and progressively refined by Transformer decoders. Each decoder contains L layers (right), consisting of position embedding, masked self-attention, stage-decoupled anchor embedding, spatio-temporal sampling, and adaptive fusion.

corresponds to a grid cell region on the BEV plane. The object queries $\mathbf{Q}_t$ and image features $\mathcal{F}_t$ are fed into a Transformer decoder $\mathcal{D}$ containing L layers. Each layer comprises position embedding, masked self-attention, stage-decoupled anchor embedding, spatial-temporal sampling, and adaptive fusion.

3.1 Progressive Query Refinement

Given the multi-frame image feature sequence $\mathcal{F}_t = \{\mathbf{F}_t, \mathbf{F}_{t-1}, \cdots, \mathbf{F}_{t-N}\}$ consisting of the current and N historical frame image features, the projection matrix sequence $\{\mathbf{P}_t, \mathbf{P}_{t-1}, \cdots, \mathbf{P}_{t-N}\}$ which projects coordinates from 3D space to the image plane, and the object queries $\{\mathbf{Q}_t, \mathbf{Q}_{t-1}, \cdots, \mathbf{Q}_{t-N}\}$, where $t-i$ denotes the timestamp and $i \in \{0, 1, \cdots, N\}$, the progressive query refinement is formulated as follows.

At keyframe $t-i$ ($0 < i \leq N$), the image feature sequence $\mathcal{F}_t$ is reordered by circularly shifting so that $\mathbf{F}_{t-i}$ is placed at the front, producing the reordered image feature sequence $\hat{\mathcal{F}}_{t-i} = \{\mathbf{F}_{t-i}, \mathbf{F}_{t-i-1}, \cdots, \mathbf{F}_{t-N}, \mathbf{F}_t, \cdots, \mathbf{F}_{t-i+1}\}$. This reordering ensures that the target keyframe $t-i$ serves as the reference frame, while treating all other keyframes as temporal context. The transformer decoder $\mathcal{D}$ takes the reordered image feature sequence $\hat{\mathcal{F}}_{t-i}$, projection matrix $\mathbf{P}_{t-i}$, and object queries $\mathbf{Q}_{t-i}$ as input to produce the refined queries:

$$\mathbf{Q}'_{t-i} = \mathcal{D}(\mathbf{Q}_{t-i}, \hat{\mathcal{F}}_{t-i}, \mathbf{P}_{t-i}). \tag{1}$$

The refined queries $\mathbf{Q}'_{t-i}$, which encode the predicted bounding box attributes (position, size, orientation, and velocity), are then transformed from the coordinate system at timestamp $t-i$ to the next timestamp $t-i+1$. Specifically, given the ego-motion matrix $\mathbf{E}_{t-i\to t-i+1}$ between the two timestamps and the time interval Δt, we first rotate the velocity by the 2D rotation component $\mathbf{E}^r_{t-i\to t-i+1}$ of $\mathbf{E}_{t-i\to t-i+1}$:

$$\tilde{\mathbf{v}}_{t-i} = {\mathbf{E}^r_{t-i\to t-i+1}}^\top \mathbf{v}_{t-i}. \tag{2}$$

Then the 3D position $\mathbf{p}_{t-i}$ is transformed by $\mathbf{E}_{t-i\to t-i+1}$ with velocity compensation:

$$\tilde{\mathbf{p}}_{t-i} = \mathbf{E}_{t-i\to t-i+1}\, \mathbf{p}_{t-i} + \tilde{\mathbf{v}}_{t-i} \cdot \Delta t, \tag{3}$$

where $\tilde{\mathbf{p}}_{t-i}$ is the compensated position in the coordinate system of timestamp $t-i+1$. Together with the rotated velocity and other bounding box attributes, they form the transformed queries $\tilde{\mathbf{Q}}_{t-i}$ in the next keyframe's coordinate system.

After the coordinate transformation, we select the top-k queries with the highest classification confidence scores from the transformed queries. This filtering step retains only the most reliable pre-

dictions for propagation, reducing noise from low-confidence detections:

$$\hat{\mathbf{Q}}_{t-i} = \mathrm{TopK}(\tilde{\mathbf{Q}}_{t-i}). \tag{4}$$

The selected queries $\hat{\mathbf{Q}}_{t-i}$ are concatenated with the object queries $\mathbf{Q}_{t-i+1}$ at the next keyframe to form the input queries for the next propagation step:

$$\bar{\mathbf{Q}}_{t-i+1} = \hat{\mathbf{Q}}_{t-i} \| \mathbf{Q}_{t-i+1}, \tag{5}$$

where $\|$ denotes concatenation. The concatenated queries are then fed into the transformer decoder:

$$\mathbf{Q}'_{t-i+1} = \mathcal{D}(\bar{\mathbf{Q}}_{t-i+1}, \hat{\mathcal{F}}_{t-i+1}, \mathbf{P}_{t-i+1}). \tag{6}$$

This process repeats recurrently until reaching the current timestamp t, where the final refined queries $\mathbf{Q}'_t$ are used for object classification and bounding box regression.

3.2 Masked Self-Attention

Denoising queries. Following DN-DETR Li et al. (2022a), we construct G groups of denoising queries by perturbing ground-truth boxes and labels. For each box, the center offsets satisfy $|\Delta x| < \lambda w/2$ and $|\Delta y| < \lambda l/2$, where λ controls the noise scale; we also randomly replace a fraction γ of the class labels.

Attention mask. To prevent information leakage while allowing regular and propagated queries to attend to each other, we design a masked attention mechanism that controls interactions among the three query types. Given denoising queries $\mathbf{Q}_{dn}$, regular queries $\mathbf{Q}_t$, and propagated queries $\hat{\mathbf{Q}}_{t-1}$, the decoder input is $\mathbf{Q}^* = \mathbf{Q}_{dn} \| \mathbf{Q}_t \| \hat{\mathbf{Q}}_{t-1}$, where $\|$ denotes concatenation. The attention mask $\mathbf{M} \in \{0, -\infty\}^{W \times W}$ is defined as:

$$\mathbf{M}_{ij} = \begin{cases} 0, & i, j \in \mathbf{Q}^g_{dn} \text{ for the same group } g, \\ 0, & i, j \in \mathbf{Q}_t \cup \hat{\mathbf{Q}}_{t-1}, \\ -\infty, & \text{otherwise}, \end{cases} \tag{7}$$

where W is the total number of queries and $\mathbf{M}_{ij} = -\infty$ indicates that query i is blocked from attending to query j. This design ensures that: (1) denoising queries attend only within their own group, preventing cross-group label leakage; (2) regular and propagated queries attend freely to each other, enabling temporal information exchange; and (3) denoising queries are isolated from regular and propagated queries, separating the auxiliary denoising task from the main detection task.

Before the masked self-attention, the masked position embedding $\bar{\mathbf{E}}_{\mathrm{pos}}$ of our stage-decoupled anchor embedding (described in Sec. 3.3) is added to the queries to encode spatial information:

$$\tilde{\mathbf{Q}}^* = \mathbf{Q}^* + \bar{\mathbf{E}}_{\mathrm{pos}}. \tag{8}$$

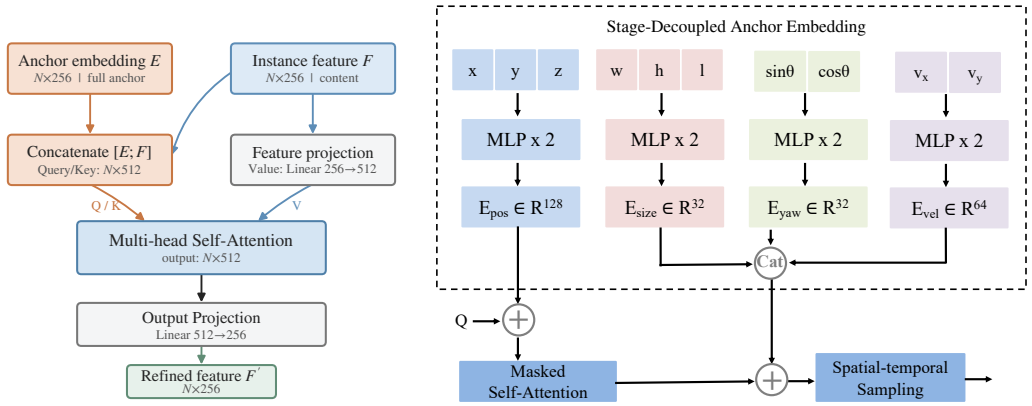
Given the pairwise BEV distance matrix $\mathbf{D} \in \mathbb{R}^{W \times W}$, where $\mathbf{D}_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ is the BEV distance between the centers of queries i and j, the masked self-attention is computed as:

$$\mathrm{Attn}(\tilde{\mathbf{Q}}^*) = \mathrm{softmax}\left(\frac{\tilde{\mathbf{Q}}^* (\tilde{\mathbf{Q}}^*)^\top}{\sqrt{d}} - \tau \mathbf{D} + \mathbf{M} \right) \tilde{\mathbf{Q}}^*, \tag{9}$$

where d is the feature dimension and τ is a learnable coefficient that controls the receptive field.

3.3 Stage-Decoupled Anchor Embedding

Several query-based detectors Liu et al. (2023a); Wang et al. (2023a); Liu et al. (2024) primarily condition object queries on position embeddings derived from the anchor center (x, y, z). Sparse4Dv3 Lin et al. (2023b) further encodes the full anchor and introduces feature–anchor decoupling. As shown in Fig. 3(a), it concatenates the anchor embedding $\mathbf{E}$ with the instance feature $\mathbf{F}$ to construct the query and key, while using the projected instance feature as the value. Thus, all anchor attributes are jointly introduced before self-attention.



(a) Feature-anchor decoupling in Sparse4D v3

(b) Stage-decoupled anchor embedding in PQR3D

Figure 3: **Comparison of anchor embedding strategies.** (a) Sparse4D v3 constructs the query and key by concatenating the anchor embedding E with the instance feature F, while deriving the value from F alone. (b) Our stage-decoupled anchor embedding injects the position embedding E_{pos} before masked self-attention, while the size, yaw, and velocity embeddings are concatenated and injected afterward, before spatio-temporal sampling. $\oplus$ denotes element-wise addition, and *Cat* denotes concatenation.

In contrast, our stage-decoupled anchor embedding injects different anchor attributes at different decoder stages, as illustrated in Fig. 3(b). Position is introduced before masked self-attention, whereas size, orientation, and velocity are introduced afterward. Specifically, given an anchor

$$\mathbf{a} = (x, y, z, w, h, l, \sin\theta, \cos\theta, v_x, v_y), \tag{10}$$

we encode its attribute groups using separate MLPs:

$$\mathbf{E}_{\text{pos}} = \text{MLP}_{\text{pos}}(x, y, z) \in \mathbb{R}^{128}, \qquad \mathbf{E}_{\text{size}} = \text{MLP}_{\text{size}}(w, h, l) \in \mathbb{R}^{32},$$
$$\mathbf{E}_{\text{yaw}} = \text{MLP}_{\text{yaw}}(\sin\theta, \cos\theta) \in \mathbb{R}^{32}, \qquad \mathbf{E}_{\text{vel}} = \text{MLP}_{\text{vel}}(v_x, v_y) \in \mathbb{R}^{64}. \tag{11}$$

Each MLP consists of two linear layers with ReLU activations. We concatenate the size, orientation, and velocity embeddings to form

$$\mathbf{E}_{\text{attr}} = [\mathbf{E}_{\text{size}} \parallel \mathbf{E}_{\text{yaw}} \parallel \mathbf{E}_{\text{vel}}] \in \mathbb{R}^{128}. \tag{12}$$

The complete anchor embedding is

$$\mathbf{E}_{\text{anchor}} = [\mathbf{E}_{\text{pos}} \parallel \mathbf{E}_{\text{attr}}] \in \mathbb{R}^{256}. \tag{13}$$

We then apply channel masking to obtain two 256-dimensional embeddings:

$$\bar{\mathbf{E}}_{\text{pos}} = [\mathbf{E}_{\text{pos}} \parallel \mathbf{0}_{128}], \qquad \bar{\mathbf{E}}_{\text{attr}} = [\mathbf{0}_{128} \parallel \mathbf{E}_{\text{attr}}]. \tag{14}$$

For $\bar{\mathbf{E}}_{\text{pos}}$, the attribute channels are masked out; for $\bar{\mathbf{E}}_{\text{attr}}$, the position channels are masked out. Both embeddings remain 256-dimensional and are therefore compatible with the query features. The masked position embedding $\bar{\mathbf{E}}_{\text{pos}}$ is added to the queries before self-attention, as in Eq. (8). After masked self-attention, the masked attribute embedding is added as a residual:

$$\mathbf{Q}' = \text{Attn}(\tilde{\mathbf{Q}}^*) + \bar{\mathbf{E}}_{\text{attr}}. \tag{15}$$

This stage-decoupled design uses position to guide query interactions while introducing geometric and motion attributes afterward, reducing interference from inconsistent attributes between regular and propagated queries.

Spatial-temporal Sampling and Adaptive Fusion. After masked self-attention and stage-decoupled anchor embedding, we follow SparseBEV Liu et al. (2023a) to aggregate multi-view multi-frame image features. The spatial-temporal sampling module generates adaptive 3D sampling offsets from query features, warps them across timestamps using ego-motion and object velocity, and projects them onto multi-view feature maps for bilinear sampling. The adaptive fusion module then fuses the sampled features via dynamic channel and point mixing. Finally, the updated queries are passed through a feed-forward network and decoded by two separate MLP heads for classification and box regression.

Table 1: Performance comparison on the nuScenes val split. ‡ indicates methods with CBGS which elongates 1 epoch into 4.5 epochs. $*$ denotes our reproduction at 24 epochs using ResNet-101 configurations adapted from the official ResNet-50 release, since neither RayDN nor Sparse4Dv3 provides ResNet-101 configurations.

Method	Backbone	Input Size	Epochs	NDS↑	mAP↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓
PETRv2 Liu et al. (2023b)	ResNet50	800 × 320	60	45.6	35.0	0.726	0.277	0.505	0.503	0.181
BEVStereo Li et al. (2023a)	ResNet50	704 × 256	90‡	50.0	37.2	0.598	0.270	0.438	0.367	0.190
BEVPoolv2 Huang & Huang (2022b)	ResNet50	704 × 256	90‡	52.6	40.6	0.572	0.275	0.463	0.275	0.188
SOLOFusion Park et al. (2023)	ResNet50	704 × 256	90‡	53.4	42.7	0.567	0.274	0.511	0.252	0.181
Sparse4Dv2 Lin et al. (2023a)	ResNet50	704 × 256	100	53.9	43.9	0.598	0.270	0.475	0.282	0.179
StreamPETR Wang et al. (2023a)	ResNet50	704 × 256	60	55.0	45.0	0.613	0.267	0.413	0.265	0.196
SparseBEV Liu et al. (2023a)	ResNet50	704 × 256	36	55.8	44.8	0.581	0.271	0.373	0.247	0.190
Sparse4Dv3 Lin et al. (2023b)	ResNet50	704 × 256	100	56.1	<u>46.9</u>	0.553	0.274	0.476	0.227	0.200
RayDN Liu et al. (2024)	ResNet50	704 × 256	60	<u>56.3</u>	<u>46.9</u>	0.579	0.264	0.433	0.256	0.187
PQR3D	ResNet50	704 × 256	24	**58.2**	**48.6**	0.576	0.263	0.353	0.224	0.191
DETR3D Wang et al. (2022)	ResNet101-DCN	1600 × 900	24	43.4	34.9	0.716	0.268	0.379	0.842	0.200
BEVFormer Li et al. (2022c)	ResNet101-DCN	1600 × 900	24	51.7	41.6	0.673	0.274	0.372	0.394	0.198
BEVDepth Li et al. (2022b)	ResNet101	1408 × 512	90‡	53.5	41.2	0.565	0.266	0.358	0.331	0.190
HoP-BEVFormer Zong et al. (2023)	ResNet101-DCN	1600 × 900	24	55.8	45.4	0.565	0.265	0.327	0.337	0.194
SOLOFusion Park et al. (2023)	ResNet101	1408 × 512	90‡	58.2	48.3	0.503	0.264	0.381	0.246	0.207
Sparse4Dv2 Lin et al. (2023a)	ResNet101	1408 × 512	100	<u>59.4</u>	<u>50.5</u>	0.548	0.268	0.348	0.239	0.184
SparseBEV Liu et al. (2023a)	ResNet101	1408 × 512	24	59.2	50.1	0.562	0.265	0.321	0.243	0.195
Sparse4Dv3* Lin et al. (2023b)	ResNet101	1408 × 512	24	56.1	48.0	0.556	0.261	0.535	0.237	0.200
RayDN* Liu et al. (2024)	ResNet101	1408 × 512	24	55.1	46.0	0.610	0.270	0.431	0.279	0.204
PQR3D	ResNet101	1408 × 512	24	**61.2**	**52.0**	0.540	0.261	0.276	0.214	0.196

4 EXPERIMENTS

4.1 IMPLEMENTATION DETAILS

BEV Flip Augmentation. We introduce a BEV flip augmentation designed for progressive query refinement. During training, we apply a mutually exclusive random flip along either the x-axis or the y-axis to the BEV coordinate system. We apply the flip only to the current frame's projection and ego-pose matrices, while keeping historical frames in their original coordinate systems. Through the relative ego-pose transformation, the inverse of the flipped ego-pose at t naturally maps past queries into the flipped coordinate system. This keeps current-frame and propagated historical queries in a consistent coordinate system, enabling stable training.

We conduct experiments with ResNet-50, ResNet-101 He et al. (2016), VoV99 Lee et al. (2019), and ViT-L Dosovitskiy et al. (2020) backbones. ResNet-50 and ResNet-101 are initialized using ImageNet Deng et al. (2009) and nuImages Caesar et al. (2020) pre-training and evaluated on the nuScenes validation set. VoV99 is initialized from DD3D Park et al. (2021), while VoV99 and ViT-L results are reported on the test set.

4.2 DATASET AND METRICS

We evaluate PQR3D on nuScenes Caesar et al. (2020), a large-scale autonomous driving benchmark with 1000 driving scenes captured by 6 surround-view cameras, 1 LiDAR, and 5 radars. The dataset is split into 700/150/150 scenes for training, validation, and testing. Each scene spans 20 seconds with key samples annotated at 2 Hz, providing 1.4M annotated 3D bounding boxes across 10 object classes. We report the nuScenes Detection Score (NDS) as the primary metric, which is a weighted combination of mean Average Precision (mAP) and five True Positive (TP) metrics: average translation error (ATE), average scale error (ASE), average orientation error (AOE), average velocity error (AVE), and average attribute error (AAE). NDS is a weighted combination of mAP and the five TP metrics (see Caesar et al. (2020) for the precise formulation), providing a holistic evaluation.

4.3 MAIN RESULTS

Validation set. As shown in Table 1, we compare PQR3D with existing methods on the nuScenes val split. With ResNet-50 as the backbone and an input resolution of 704×256, PQR3D achieves 58.2 NDS and 48.6 mAP, surpassing the previous best method RayDN by $+1.9$ NDS and $+1.7$ mAP. Notably, PQR3D is trained for only 24 epochs, while RayDN requires 60 epochs to reach its reported performance. Compared with SparseBEV, which uses a relatively short 36-epoch schedule, PQR3D achieves further gains of $+2.4$ NDS and $+3.8$ mAP with only 24 training epochs. These

Table 2: Performance comparison on the nuScenes test split. † indicates configurations that use future frames. ‡ indicates training with CBGS, for which one nominal epoch corresponds to approximately 4.5 standard epochs. $*$ indicates the use of test-time augmentation.

Method	Backbone	Input Size	Epochs	NDS↑	mAP↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓
PETRv2 Liu et al. (2023b)	VoV99	1600 × 640	24	58.2	49.0	0.561	0.243	0.361	0.343	0.120
Sparse4D Lin et al. (2022)	VoV99	1600 × 640	48	59.5	51.1	0.533	0.263	0.369	0.317	0.124
MV2D-T Wang et al. (2023b)	VoV99	1600 × 640	72	59.6	51.1	0.525	0.243	0.357	0.357	0.120
BEVDepth Li et al. (2022b)	VoV99	1600 × 640	$90^{\ddagger}$	60.0	50.3	0.445	0.245	0.378	0.320	0.126
HoP-BEVFormer Zong et al. (2023)	VoV99	1600 × 640	24	60.3	51.7	0.501	0.245	0.346	0.362	0.105
BEVStereo Li et al. (2023a)	VoV99	1600 × 640	$90^{\ddagger}$	61.0	52.5	0.431	0.246	0.358	0.357	0.138
CAPE-T Xiong et al. (2023)	VoV99	1600 × 640	$24^{\ddagger} + 60$	61.0	52.5	0.503	0.242	0.361	0.306	0.114
FB-BEV Li et al. (2023b)	VoV99	1600 × 640	30	62.4	53.7	0.439	0.250	0.358	0.270	0.128
StreamPETR Wang et al. (2023a)	VoV99	1600 × 640	60	63.6	55.0	0.479	0.239	0.317	0.241	0.119
HENet Xia et al. (2024)	VoV99 & ResNet50	1152 × 640 & 704 × 256	$54^{\ddagger}$	63.8	57.5	0.432	0.242	0.368	0.320	0.129
SparseBEV† Liu et al. (2023a)	VoV99	1600 × 640	24	<u>67.5</u>	<u>60.3</u>	0.425	0.239	0.311	0.172	0.116
PQR3D†	VoV99	1600 × 640	24	**68.0**	**61.0**	0.406	0.239	0.318	0.168	0.118
StreamPETR Wang et al. (2023a)	ViT-L	1600 × 800	24	67.6	62.0	0.470	0.241	0.258	0.236	0.134
HoP-BEVDet4D-Depth† Zong et al. (2023)	ViT-L	1600 × 640	$36 + 5^{\ddagger}$	68.5	62.4	0.367	0.249	0.354	0.171	0.131
RayDN Liu et al. (2024)	ViT-L	1600 × 800	24	68.6	63.1	0.437	0.235	0.283	0.220	0.120
RoPETR Ji et al. (2025)	ViT-L	1600 × 640	24	69.0	61.9	0.396	0.254	0.249	0.163	0.129
SparseBEV† Liu et al. (2023a)	ViT-L	1600 × 640	24	70.2	63.5	0.402	0.235	0.237	0.154	0.129
RoPETR-e* Ji et al. (2025)	ViT-L	1600 × 900	24	<u>70.9</u>	<u>64.8</u>	0.379	0.227	0.248	0.173	0.125
HENet++† Xia et al. (2025)	ViT-L & VoV99	1600 × 640	$54^{\ddagger}$	70.7	64.5	0.402	0.235	0.237	0.155	0.129
PQR3D†	ViT-L	1600 × 640	24	**71.6**	**64.9**	0.383	0.235	0.210	0.142	0.115

Table 3: Ablation studies on the nuScenes validation split using ResNet-50, 704×256 input resolution, and 24 training epochs. (a) Effect of each component. PQR: progressive query refinement; MSA: masked self-attention; SDAE: stage-decoupled anchor embedding; BF: BEV flip. (b) Comparison of temporal propagation strategies under the same detector and training settings. Vanilla uses no query propagation, whereas Sequential uses a StreamPETR-style strategy that carries cached queries from one frame to the next along the video sequence.

(a) Component ablation

PQR	MSA	SDAE	BF	NDS↑	mAP↑
✗	✗	✗	✗	55.5	45.3
✓	✗	✗	✗	56.6	46.5
✓	✓	✗	✗	57.0	47.2
✓	✓	✓	✗	57.7	48.2
✓	✓	✓	✓	**58.2**	**48.6**

(b) Propagation strategy

Strategy	NDS↑	mAP↑
Vanilla	55.5	45.3
Sequential	55.7	45.7
PQR (ours)	**56.6**	**46.5**

results demonstrate that PQR3D consistently outperforms prior state-of-the-art methods under the ResNet-50 setting while requiring fewer training epochs.

When scaling to ResNet-101, RayDN and Sparse4Dv3 report results from models trained for 60 and 100 epochs, respectively, but do not release the corresponding configuration files. For a fair comparison under the 24-epoch setting, we adapt their open-source ResNet-50 configurations to ResNet-101 and report the reproduced results. As shown in Table 1, PQR3D achieves an NDS of 61.2 and an mAP of 52.0, outperforming the previous best model Sparse4Dv2, by $+1.8$ NDS and $+1.5$ mAP, despite being trained for only 24 epochs, whereas Sparse4Dv2 is trained for 100 epochs. Compared with SparseBEV, which uses the same 24-epoch setting, PQR3D further improves performance by $+2.0$ NDS and $+1.9$ mAP. These results demonstrate that PQR3D scales effectively to larger backbones and higher input resolutions, while maintaining strong training efficiency.

Test set. As shown in Table 2, we compare PQR3D with existing camera-only methods on the nuScenes test split. With VoV99, PQR3D achieves state-of-the-art performance among methods using the same backbone, reaching 68.0 NDS and 61.0 mAP. Under the same 1600×640 input resolution and 24-epoch training schedule, PQR3D outperforms SparseBEV by 0.5 NDS and 0.7 mAP points. With ViT-L, PQR3D achieves 71.6 NDS and 64.9 mAP, establishing new state-of-the-art performance. Compared with SparseBEV under the same backbone, input resolution, and training schedule, PQR3D improves both NDS and mAP by 1.4 points. It also surpasses RoPETR-e by 0.7 NDS and 0.1 mAP, despite RoPETR-e using a higher 1600×900 resolution and test-time augmentation.

4.4 Ablation Study

Component ablation. As shown in Table 3(a), we evaluate the contribution of each proposed component by incrementally adding it to the baseline. Progressive Query Refinement (PQR) provides the

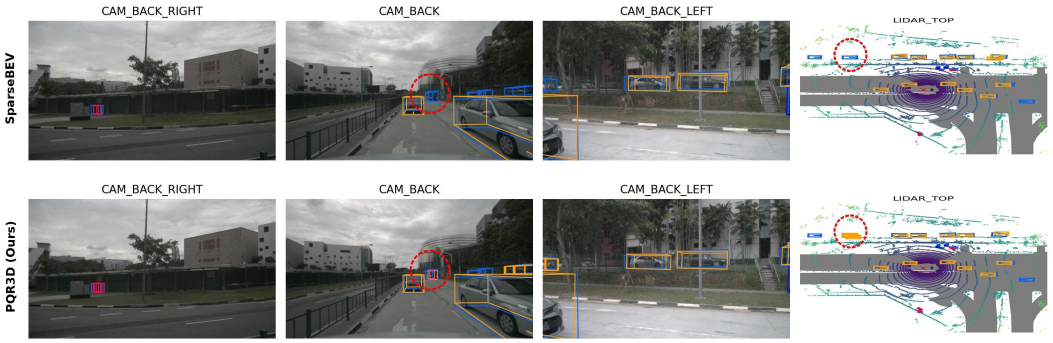


Figure 4: Qualitative comparison of SparseBEV (top) and PQR3D (bottom) on the nuScenes val set. Ground-truth boxes are shown in blue and predictions are class-colored (orange for car, crimson for bicycle, etc.). Red dashed circles mark regions of interest.

largest single improvement of $+1.1$ NDS and $+1.2$ mAP, demonstrating that progressively aligning and refining high-confidence queries toward the target frame effectively leverages information from earlier frames. Adding masked self-attention (MSA) yields a further $+0.4$ NDS and $+0.7$ mAP by isolating denoising queries while preserving interactions between regular and propagated queries. Stage-decoupled anchor embedding (SDAE) brings an additional $+0.7$ NDS and $+1.0$ mAP by applying the position embedding before self-attention and injecting the remaining anchor attributes afterward, reducing conflicts among temporal queries. Finally, BEV flip augmentation (BF) brings $+0.5$ NDS and $+0.4$ mAP, providing spatial regularization compatible with temporal propagation. Overall, the four components improve the baseline by $+2.7$ NDS and $+3.3$ mAP, with complementary gains from each component.

Propagation strategy. As shown in Table 3(b), we compare three propagation strategies under the same settings. Vanilla uses no query propagation, whereas the StreamPETR-style Sequential baseline uses sequential video sampling and carries cached queries from one frame to the next, yielding only $+0.2$ NDS and $+0.4$ mAP. In contrast, PQR independently processes each temporal window and progressively refines queries from earlier frames toward the target frame, improving Vanilla by $+1.1$ NDS and $+1.2$ mAP. PQR also outperforms Sequential by $+0.9$ NDS and $+0.8$ mAP, demonstrating the advantage of window-based progressive refinement over frame-by-frame propagation.

4.5 Visualization

Figure 4 shows a qualitative comparison between the baseline SparseBEV and our PQR3D on the nuScenes validation set. Within the highlighted region marked by the red dashed circle, SparseBEV fails to detect a distant vehicle, whereas PQR3D successfully recovers it by progressively refining object queries with temporal evidence from preceding frames. This improvement is visible in both the rear-camera view and the LiDAR BEV visualization, demonstrating that progressive query refinement helps recover objects with weak visual evidence in the current frame.

5 Conclusion

We present PQR3D, a progressive query refinement framework for multi-view 3D object detection. Within each reference-conditioned temporal window, PQR3D progressively refines object queries from earlier keyframes toward the target frame using ego-pose and velocity compensation. A masked self-attention mechanism is designed for the interactions among denoising, regular, and propagated queries to prevent information leakage while allowing regular and propagated queries to attend to each other. Furthermore, a stage-decoupled anchor embedding injects geometric and motion attributes without introducing conflicts during attention computation. Extensive experiments on nuScenes demonstrate that PQR3D achieves state-of-the-art performance across multiple backbone configurations. Future work includes extending PQR3D to broader 3D perception tasks such as BEV segmentation and occupancy prediction.

AI USE STATEMENT

Generative AI tools were used for language editing, including grammar and spelling correction, word choice, and readability improvement; to assist in reviewing source code for potential errors; and to assist in creating Figs. 1, 3(a), 4, and 5. Figs. 2 and 3(b) were created entirely by the authors. All AI-assisted text, code-review suggestions, and visual materials were reviewed, revised, and verified by the authors. All research ideas and methods were developed by the authors, who also designed and conducted the experiments, analyzed the results, and drew the conclusions. The authors take full responsibility for the final content of this paper.

ETHICS STATEMENT

This work uses the publicly available nuScenes dataset and collects no new human-subject data. Improved 3D object detection may enhance autonomous-driving safety, but perception failures could lead to unsafe decisions. PQR3D is a research prototype, and real-world deployment requires extensive validation, system redundancy, and fail-safe mechanisms.

REPRODUCIBILITY STATEMENT

Implementation and training details are provided in Sec. 4.1 and Appendix A.1. During review, our anonymous repository at `https://anonymous.4open.science/r/pqr3d-review-B5E8` provides the source code, configuration files, environment and package-installation instructions, training and evaluation commands, trained checkpoints, and training logs. These resources support reproduction of the reported nuScenes experiments and results. The code and models will be publicly released on GitHub after the review process.

REFERENCES

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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11621–11631, 2020.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 248–255, 2009.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Chengjian Feng, Zequn Jie, Yujie Zhong, Xiangxiang Chu, and Lin Ma. Aedet: Azimuth-equivariant detector for multi-view 3d object detection. *arXiv preprint arXiv:2211.12501*, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Junjie Huang and Guan Huang. Bevdet4d: Exploit temporal cues in multi-camera 3d object detection. *arXiv preprint arXiv:2203.17054*, 2022a.

Junjie Huang and Guan Huang. Bevpoolv2: A cutting-edge implementation of bevdet toward deployment. *arXiv preprint arXiv:2211.17111*, 2022b.

Junjie Huang, Guan Huang, Zheng Zhu, and Dalong Du. Bevdet: High-performance multi-camera 3d object detection in bird-eye-view. *arXiv preprint arXiv:2112.11790*, 2021.

Hang Ji, Tao Ni, Xufeng Huang, Zhan Shi, Tao Luo, Xin Zhan, and Junbo Chen. Ropetr: Improving temporal camera-only 3d detection by integrating enhanced rotary position embedding. *arXiv preprint arXiv:2504.12643*, 2025.

Xiaohui Jiang, Shuailin Li, Yingfei Liu, Shihao Wang, Fan Jia, Tiancai Wang, Lijun Han, and Xiangyu Zhang. Far3d: Expanding the horizon for surround-view 3d object detection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 2561–2569, 2024.

Youngwan Lee, Joong-won Hwang, Sangrok Lee, Yuseok Bae, and Jongyoul Park. An energy and gpu-computation efficient backbone network for real-time object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 0–0, 2019.

Feng Li, Hao Zhang, Shilong Liu, Jian Guo, Lionel M Ni, and Lei Zhang. Dn-detr: Accelerate detr training by introducing query denoising. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13619–13627, 2022a.

Yinhao Li, Zheng Ge, Guanyi Yu, Jinrong Yang, Zengran Wang, Yukang Shi, Jianjian Sun, and Zeming Li. Bevdepth: Acquisition of reliable depth for multi-view 3d object detection. *arXiv preprint arXiv:2206.10092*, 2022b.

Yinhao Li, Han Bao, Zheng Ge, Jinrong Yang, Jianjian Sun, and Zeming Li. Bevstereo: Enhancing depth estimation in multi-view 3d object detection with dynamic temporal stereo. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 1486–1494, 2023a.

Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Yu Qiao, and Jifeng Dai. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In *European conference on computer vision*, pp. 1–18, 2022c.

Zhiqi Li, Zhiding Yu, Wenhai Wang, Anima Anandkumar, Tong Lu, and Jose M Alvarez. Fb-bev: Bev representation from forward-backward view transformations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6919–6928, 2023b.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

Xuewu Lin, Tianwei Lin, Zixiang Pei, Lichao Huang, and Zhizhong Su. Sparse4d: Multi-view 3d object detection with sparse spatial-temporal fusion. *arXiv preprint arXiv:2211.10581*, 2022.

Xuewu Lin, Tianwei Lin, Zixiang Pei, Lichao Huang, and Zhizhong Su. Sparse4d v2: Recurrent temporal fusion with sparse model. *arXiv preprint arXiv:2305.14018*, 2023a.

Xuewu Lin, Zixiang Pei, Tianwei Lin, Lichao Huang, and Zhizhong Su. Sparse4d v3: Advancing end-to-end 3d detection and tracking. *arXiv preprint arXiv:2311.11722*, 2023b.

Feng Liu, Tengteng Huang, Qianjing Zhang, Haotian Yao, Chi Zhang, Fang Wan, Qixiang Ye, and Yanzhao Zhou. Ray denoising: Depth-aware hard negative sampling for multi-view 3d object detection. In *European conference on computer vision*, 2024.

Haisong Liu, Yao Teng, Tao Lu, Haiguang Wang, and Limin Wang. Sparsebev: High-performance sparse 3d object detection from multi-camera videos. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 18580–18590, 2023a.

Yingfei Liu, Tiancai Wang, Xiangyu Zhang, and Jian Sun. Petr: Position embedding transformation for multi-view 3d object detection. In *European conference on computer vision*, pp. 531–548, 2022.

Yingfei Liu, Junjie Yan, Fan Jia, Shuailin Li, Qi Gao, Tiancai Wang, Xiangyu Zhang, and Jian Sun. Petrv2: A unified framework for 3d perception from multi-camera images. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3262–3272, 2023b.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International conference on learning representations*, 2017.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International conference on learning representations*, 2019.

Dennis Park, Rares Ambrus, Vitor Guizilini, Jie Li, and Adrien Gaidon. Is pseudo-lidar needed for monocular 3d object detection? In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3142–3152, 2021.

Jinhyung Park, Chenfeng Xu, Shijia Yang, Kurt Keutzer, Kris Kitani, Masayoshi Tomizuka, and Wei Zhan. Time will tell: New outlooks and a baseline for temporal multi-view 3d object detection. In *International conference on learning representations*, 2023.

Jonah Philion and Sanja Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *European conference on computer vision*, pp. 194–210, 2020.

Shihao Wang, Yingfei Liu, Tiancai Wang, Ying Li, and Xiangyu Zhang. Exploring object-centric temporal modeling for efficient multi-view 3d object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3621–3631, 2023a.

Yue Wang, Vitor Campagnolo Guizilini, Tianyuan Zhang, Yilun Wang, Hang Zhao, and Justin Solomon. Detr3d: 3d object detection from multi-view images via 3d-to-2d queries. In *Conference on robot learning*, pp. 180–191, 2022.

Zitian Wang, Zehao Huang, Jiahui Fu, Naiyan Wang, and Si Liu. Object as query: Lifting any 2d object detector to 3d detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3791–3800, 2023b.

Zhongyu Xia, Zhiwei Lin, Xinhao Wang, Yongtao Wang, Yun Xing, Shengxiang Qi, Nan Dong, and Ming-Hsuan Yang. Henet: Hybrid encoding for end-to-end multi-task 3d perception from multi-view cameras. In *European conference on computer vision*, 2024.

Zhongyu Xia et al. Henet++: Hybrid encoding and multi-task learning for 3d perception and end-to-end autonomous driving. *arXiv preprint arXiv:2511.07106*, 2025.

Kaixin Xiong, Shi Gong, Xiaoqing Ye, Xiao Tan, Ji Wan, Errui Ding, Jingdong Wang, and Xiang Bai. Cape: Camera view position embedding for multi-view 3d object detection. In *2023 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 21570–21579. IEEE, 2023.

Chenyu Yang, Yuntao Chen, Hao Tian, Chenxin Tao, Xizhou Zhu, Zhaoxiang Zhang, Gao Huang, Hongyang Li, Yu Qiao, Lewei Lu, et al. Bevformer v2: Adapting modern image backbones to bird's-eye-view recognition via perspective supervision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 17830–17839, 2023.

Zhuofan Zong, Dongzhi Jiang, Guanglu Song, Zeyue Xue, Jingyong Su, Hongsheng Li, and Yu Liu. Temporal enhanced training of multi-view 3d object detector via historical object prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3781–3790, 2023.

A Additional Experiments and Analysis

A.1 Implementation Details

Unless otherwise specified, models used for benchmark comparison and ablation studies are trained for 24 epochs using AdamW Loshchilov & Hutter (2019) with a batch size of 8, a base learning rate of 2×10^{-4}, and cosine annealing Loshchilov & Hutter (2017). The Transformer decoder contains $L = 6$ layers.

The validation configurations use the current frame and eight preceding frames sampled at approximately 0.5-s intervals. Test configurations marked with † additionally use future frames and therefore correspond to offline rather than causal online inference.

We use focal loss Lin et al. (2017) for classification and L1 loss for bounding-box regression. For query denoising, we set $G = 10$ and use a noise scale of $\lambda = 0.4$, following SparseBEV Liu et al. (2023a). Unless otherwise specified, PQR3D uses two refinement steps with $k = 256$ propagated queries per step.

ResNet-50 and ResNet-101 models are trained on $8\times$ NVIDIA RTX 4090 GPUs, while VoV99 and ViT-L models are trained on $8\times$ NVIDIA A100 GPUs. The lightweight 625-query configuration used for the efficiency comparison is trained for 48 epochs.

A.2 Inference Speed

The standard PQR3D configuration uses 900 queries and is trained for 24 epochs. For the efficiency comparison, we additionally evaluate a lightweight configuration with 625 queries trained for 48 epochs. As shown in Figure 5, all methods are evaluated on a single NVIDIA RTX 3090 GPU under the same inference settings. The lightweight PQR3D achieves 58.5 NDS and 48.1 mAP while running at approximately 20 FPS. At comparable throughput, it outperforms Sparse4Dv3 by 2.4 NDS points. Although RayDN and SparseBEV run faster, PQR3D improves NDS by 2.2 and 2.7 points, respectively. These results demonstrate that PQR3D provides a favorable balance between detection accuracy and inference speed.

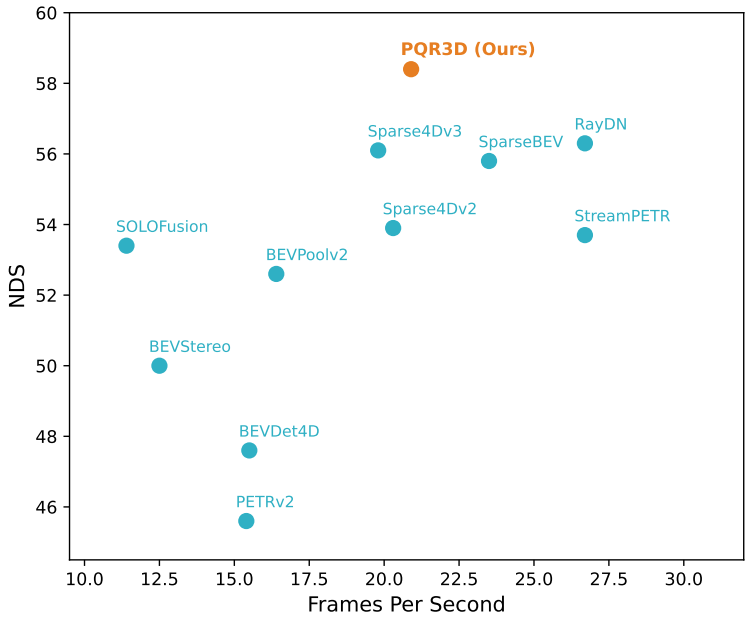


Figure 5: Speed–accuracy comparison of PQR3D with existing methods on the nuScenes validation set in terms of frames per second (FPS) and NDS.

A.3 Additional Ablation Studies

We further study the number of progressive refinement steps, the number of propagated queries, and the anchor embedding design.

Table 4: Effect of the number of progressive refinement steps on detection accuracy and GPU memory usage.

Steps	NDS↑	mAP↑	Mem. (GB)↓
0	55.5	45.3	8.1
1	56.2	46.1	8.3
2	56.6	**46.5**	8.7
3	**56.8**	46.4	9.3

Number of refinement steps. As shown in Table 4, one progressive refinement step improves the baseline by $+0.7$ NDS and $+0.8$ mAP. Using two steps provides a further gain of $+0.4$ NDS and $+0.4$ mAP by incorporating an additional historical keyframe. A third step yields only $+0.2$ NDS, decreases mAP by 0.1, and increases memory usage from 8.7 to 9.3 GB, indicating diminishing returns from additional refinement. We therefore use two refinement steps by default to balance detection accuracy and memory usage.

Table 5: Ablation studies on propagated query count and anchor embedding design. (a) Number of propagated queries k. (b) Anchor embedding design. Pos: position-only embedding.

(a) Propagated query count k

k	NDS↑	mAP↑	Mem. (GB)↓
0	55.5	45.3	8.1
128	56.3	46.0	8.6
256	56.6	**46.5**	8.7
512	**56.7**	46.2	8.9

(b) Anchor embedding design

Variant	NDS↑	mAP↑
Pos	57.0	47.2
Coupled	57.2	46.9
Stage-decoupled	**57.7**	**48.2**

Propagated query count. As shown in Table 5(a), setting $k = 128$ improves the baseline by $+0.8$ NDS and $+0.7$ mAP, showing that even a small set of propagated queries is beneficial. Increasing k to 256 provides further gains of $+0.3$ NDS and $+0.5$ mAP by transferring more high-confidence object queries between refinement steps. Increasing k to 512, however, improves NDS by only 0.1, decreases mAP by 0.3, and adds 0.2 GB of memory. We therefore use $k = 256$ by default.

Anchor embedding design. As shown in Table 5(b), the coupled design improves NDS by 0.2 but decreases mAP by 0.3 compared with the position-only baseline. This suggests that injecting all anchor attributes before self-attention can introduce conflicts between regular and propagated queries. In contrast, our stage-decoupled design injects position before self-attention and adds size, orientation, and velocity afterward. It achieves 57.7 NDS and 48.2 mAP, improving the position-only baseline by 0.7 NDS and 1.0 mAP. These results show that stage-wise attribute injection enriches query representations while reducing conflicts among temporal queries.

A.4 Additional Qualitative Results

Figure 6 presents another qualitative comparison on the nuScenes validation set. In the highlighted region, SparseBEV produces a poorly localized bounding box for a partially occluded truck, whereas PQR3D provides more accurate localization through progressive query refinement within the temporal window. This example shows that progressive query refinement improves both object recall and localization accuracy under occlusion.

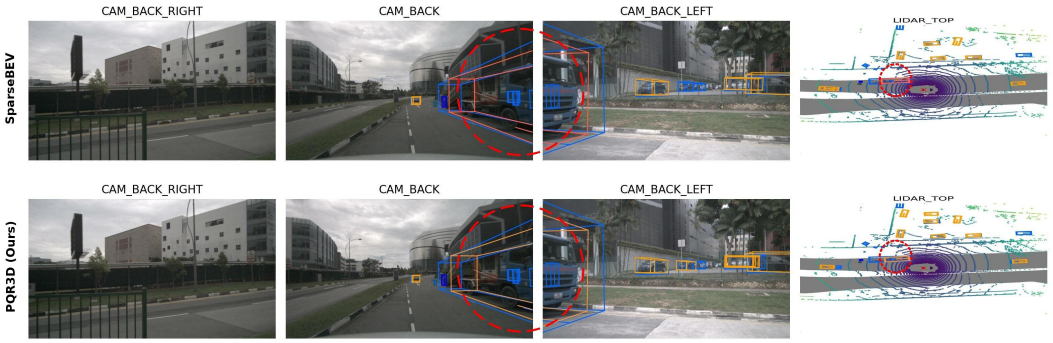


Figure 6: Qualitative comparison of SparseBEV (top) and PQR3D (bottom) on the nuScenes validation set. Ground-truth boxes are shown in blue, and predictions are class-colored. Red dashed circles indicate regions of interest.