QUERY ANYTIME: FIND WHEN BEFORE WHERE FOR ONLINE 4D POINT CLOUD SEGMENTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Query-driven 4D point cloud segmentation aims to segment target instances in dynamic 3D sequences based on natural language queries. Existing methods largely follow a query-first offline setting, where the query is given in advance and the input sequence is temporally bounded around the queried target. However, in real-world interactions, queries may arrive at arbitrary times while 3D observations continue to evolve. In such an online scenario, the query and its relevant observations are no longer temporally aligned in advance, requiring the model to establish their correspondence dynamically. We summarize this challenge as "*when* before *where*": the model must first identify *when* the queried target is present in the evolving sequence, before determining *where* it should be segmented. Based on this principle, we propose **QueryAnytime**. For *when*, a Searchable Evidence Tokenization (SET) module is proposed to build a compact, query-independent evidence space for efficient anytime query grounding without revisiting dense history. For *where*, a TrackState module is designed to update a structured semantic-geometric target state, which maintains target identity and spatial continuity for instance-consistent segmentation over time. We further introduce **Online4DKITTI**, a dedicated benchmark for online query-driven 4D segmentation with arbitrary query timing and fine-grained 3D instance annotations. Experiments demonstrate that QueryAnytime substantially outperforms existing baselines in segmentation accuracy while maintaining acceptable inference latency. Code and benchmark will be released at: https://anonymous.4open.science/r/QueryAnytime-E979/

1 INTRODUCTION

Perceiving and understanding the surrounding environment is fundamental for intelligent systems to interact with the physical world. Toward this goal, early studies introduced static 3D query-driven point cloud segmentation (Huang et al., 2021; Wu et al., 2024; He et al., 2024). These methods use language to guide mask prediction in static point cloud scenes. To move beyond static scenes, query-driven 4D segmentation (Li et al., 2026) was introduced to extend language-guided perception into the temporal dimension, aiming to identify targets across 3D frames.

Despite this progress, existing query-driven 4D segmentation (Li et al., 2026) still follows a query-first offline setting. The query is provided before sequence processing, while the input sequence (3D frames) is already temporally clipped around the queried target. This setup largely removes the need to determine which frames are temporally relevant to the query, allowing existing methods to focus primarily on spatial segmentation.

However, real-world interaction couples two dynamics that are not jointly considered in the existing formulation: queries may arrive at arbitrary times, while 3D observations continue to arrive as an ongoing stream. In autonomous driving, for example, a user may issue a query only after a vehicle has attracted attention. By that time, some relevant 3D observations (frames) may already have passed, while new observations continue to arrive and the queried target may eventually disappear. In such an online scenario, neither the query arrival time nor the target-relevant range of the sequence can be known beforehand (Fig. 1).

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.

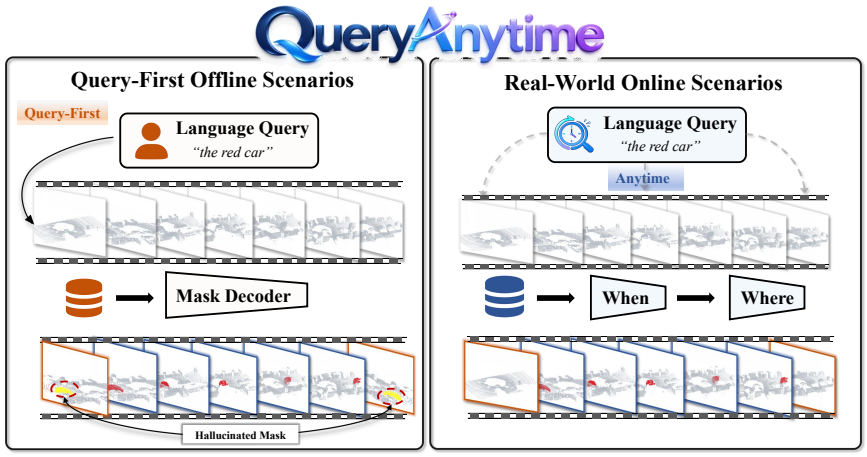


Figure 1: The top and bottom rows show input point cloud streams and predicted masks, respectively. Blue-outlined frames indicate target presence, orange-outlined frames indicate target absence. Red regions denote predicted masks and yellow regions denote hallucinations. *Left:* Query-first offline methods decode every frame, hallucinating masks when the target is absent. *Right:* For queries arriving at arbitrary times, QueryAnytime follows *when before where*: it first identifies target-present frames in the observed and incoming stream, then decodes masks only for those frames.

Motivated by this gap, we propose to study Online Query-driven 4D Point Cloud Segmentation. In this task, given a continuous stream of 3D observations and a language query arriving at an arbitrary time, the model must dynamically identify the query-relevant temporal span and segment the referred instance within it. The key challenge is that temporal correspondence between the query and the 3D sequence is no longer predefined, but must be dynamically inferred by the model itself. We summarize this shift as "*when* before *where*": the model must first determine *when* the queried target is present in the evolving stream, and then determine *where* it lies within the relevant observations.

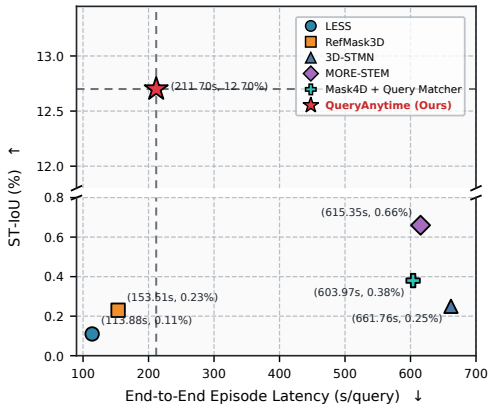


Figure 2: **Accuracy–efficiency trade-off on Online4DKITTI.** QueryAnytime achieves substantially higher ST-IoU with moderate end-to-end episode latency. Original baseline inference rules are shown here; their NTC adaptations are reported in Table 1. The vertical axis is discontinuous for readability.

Nevertheless, this formulation poses two challenging requirements. For *when*, the query is unknown while historical observations are processed. The model must learn how to preserve query-independent evidence that can later support temporal grounding, while avoiding costly reprocessing of the dense history. For *where*, once the target is temporally grounded, the model should consistently segment the same physical instance across relevant observations despite facing motion, viewpoint changes, and occlusion.

To address these requirements, we propose **QueryAnytime**, an online query-driven 4D point cloud segmentation framework that supports queries arriving at arbitrary times. For *when*, we introduce a Searchable Evidence Tokenization (SET) module. As 3D observations arrive over time, SET continuously compresses them into compact, query-agnostic tokens that preserve rich visual evidence for future anytime queries. When a query becomes available, the model efficiently searches these compact tokens to identify and retrieve temporally relevant observations. For *where*, we introduce a TrackState module, which maintains a persistent target state over the evolving stream by jointly updating semantic and geometric representations of the queried instance. The semantic representations preserve the identity of the referred instance, while the geometric representations capture its spatial continuity as the scene evolves. This structured target state guides the model to consistently localize and segment the same physical instance under motion, viewpoint changes, and occlusion.

To further support this new setting, we construct **Online4DKITTI** based on SemanticKITTI through a semi-automated annotation pipeline (Behley et al., 2019), providing 150,383 query–instance–frame 3D mask records. To instantiate the query-anytime setting, we extend each input sequence beyond the target-relevant span and vary the query arrival time throughout the sequence, thereby decoupling when a query is issued from when its target evidence appears. Online4DKITTI therefore provides a dedicated benchmark for evaluating whether a model can recover the complete target-relevant segmentation under arbitrary query arrival times, including relevant historical observations, while avoiding predictions outside the target-relevant span.

Extensive experiments on Online4DKITTI demonstrate that QueryAnytime consistently outperforms existing baselines under arbitrary query arrival times, achieving superior segmentation accuracy while maintaining acceptable inference latency (Fig. 2). Our contributions can be summarized as follows:

❶ We introduce Online Query-driven 4D Point Cloud Segmentation, relaxing prior query-first and target-always-present assumptions for more realistic interactive perception.

❷ We propose **QueryAnytime**, with SET for searchable query-agnostic evidence and TrackState for instance-consistent segmentation over time.

❸ We construct **Online4DKITTI**, decoupling query arrival from target visibility and extending sequences with irrelevant observations for arbitrary-time online evaluation.

2 Related Work

Query-driven 3D and 4D point-cloud segmentation. Query-driven point-cloud segmentation grounds natural-language descriptions in 3D scenes and predicts point-level masks. Early methods such as ScanRefer and ReferIt3D established language-guided 3D object grounding (Chen et al., 2020; Achlioptas et al., 2020). Subsequent methods improved fine-grained spatial alignment through superpoint–text matching, geometry-aware language fusion, and single-stage mask prediction (Wu et al., 2024; He & Ding, 2024; Liu et al., 2024). MORE-STEM further extends query-driven segmentation to dynamic 4D point clouds by modeling cross-frame text–visual alignment and spatio-temporal consistency (Li et al., 2026). However, these methods assume that the query and a query-relevant sequence are available before inference, leaving online temporal grounding unexplored.

Online visual grounding and query-agnostic memory. Online referring video methods study language-guided segmentation over evolving visual streams (Wu et al., 2022; 2023). Recent online video understanding works further consider long streams, arbitrary interaction timing, and efficient evidence retention (Huang et al., 2025; Niu et al., 2025; Azad et al., 2026). Query-agnostic memory and compressed KV representations provide compact interfaces for retrieving information from continuously arriving observations (Yang et al., 2025; Kim et al., 2025). These studies mainly focus on 2D video or language-model interaction and do not address point-level temporal grounding in multi-modal 3D streams. QueryAnytime introduces searchable evidence tokens that retain query-relevant 3D information before a query arrives.

Temporal instance modeling in 4D point clouds. 4D point-cloud segmentation maintains physical instances across consecutive LiDAR observations through spatio-temporal feature aggregation, instance association, and query propagation (Aygun et al., 2021; Marcuzzi et al., 2022). Mask4D propagates decoder queries across frames to obtain temporally consistent language-agnostic tracks (Marcuzzi et al., 2023). MORE-STEM introduces query-conditioned temporal modeling for language-guided 4D segmentation (Li et al., 2026). These approaches primarily preserve model-defined instance identities or dense temporal features. QueryAnytime instead maintains a compact TrackState that couples query-specific semantic identity with geometric continuity, enabling efficient instance-consistent segmentation after temporal grounding.

3 Method

3.1 Problem Formulation

We consider a multimodal stream $\mathcal{X}_t = (\mathcal{P}_t, \mathcal{I}_t)$, where $\mathcal{P}_t = \{(\mathbf{x}_{t,n}, \mathbf{f}_{t,n})\}_{n=1}^{N_t}$ is a LiDAR scan and $\mathcal{I}_t$ its synchronized RGB image. A natural-language query q arrives at time τ_q and refers to

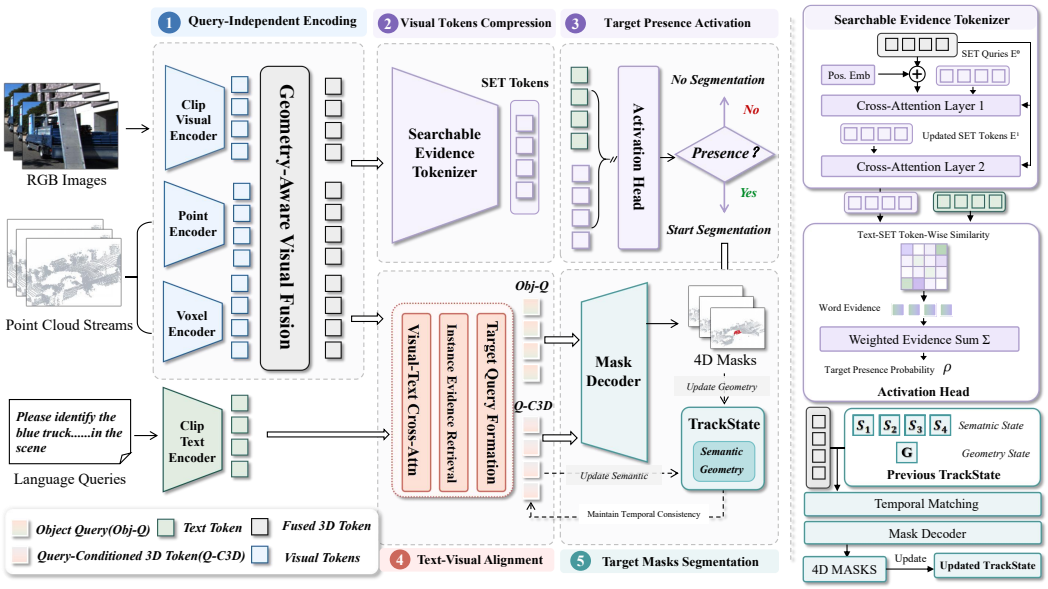


Figure 3: **QueryAnytime overview.** Each RGB–LiDAR frame is encoded once into dense Fused 3D Tokens and compact SET tokens. Upon query arrival, SET identifies *when* the target is present in the history and incoming stream, triggering coarse-to-fine grounding and mask decoding only for activated frames. TrackState maintains semantic identity and geometric continuity for consistent segmentation over time.

one physical instance whose presence may vary across the stream. Let $\mathbf{m}_t^q \in \{0,1\}^{N_t}$ denote its point-wise mask, with an all-zero mask indicating absence, and $a_t^q = \mathbb{I}[\|\mathbf{m}_t^q\|_1 > 0]$ its presence label.

Let $\mathcal{M}_t = \mathcal{U}_\theta(\mathcal{M}_{t-1}, \mathcal{X}_t)$, with $\mathcal{M}_0 = \emptyset$, denote the query-independent memory updated once per observation. At query arrival, the model searches this cached history and then processes subsequent frames online:

$$\begin{aligned} \big(\widehat{\mathcal{Y}}_{1:\tau_q}^q, \mathcal{T}_{\tau_q}^q\big) &= \mathcal{F}_\theta^{\text{hist}}(q; \mathcal{M}_{\tau_q}), \\ (\hat{a}_t^q, \hat{\mathbf{m}}_t^q, \mathcal{T}_t^q) &= \mathcal{F}_\theta^{\text{online}}(q; \mathcal{M}_t, \mathcal{T}_{t-1}^q), \qquad t > \tau_q. \end{aligned} \tag{1}$$

Here $\widehat{\mathcal{Y}}_{1:\tau_q}^q = \{(\hat{a}_s^q, \hat{\mathbf{m}}_s^q)\}_{s=1}^{\tau_q}$ is the historical output, $\hat{a}_t^q$ and $\hat{\mathbf{m}}_t^q$ are the binary presence prediction and mask, and $\mathcal{T}_t^q$ is the TrackState.

3.2 Overview

QueryAnytime separates online query serving into query-independent evidence tokenization and query-conditioned target decoding (Fig. 3). Before query arrival, each frame is encoded once into dense Fused 3D Tokens and compact SET Tokens. Together they form a Hierarchical memory for efficient retrieval and detailed spatial prediction.

After query arrival, the activation head matches the query against historical and incoming SET Tokens to determine *when* the target is present. Activated frames retrieve their dense features, from which text–visual grounding constructs Q-C3D Tokens and an Obj-Q to predict *where* the target is. TrackState, comprising semantic and geometric components, preserves the referred instance across activated frames, conditions each activated prediction, and is propagated across inactive frames. We learn these capabilities through three successive stages.

3.3 Query-Independent Evidence Memory

Each incoming frame is represented by compact SET Tokens paired with dense spatial features.

Query-independent scene encoding. A CLIP visual encoder, point encoder, and sparse voxel encoder produce image, point, and voxel features. Projected image features are injected into valid 3D

points through a learned gate, followed by bidirectional image–3D cross-attention. We denote the fused point and voxel outputs by $\widetilde{\mathbf{H}}_t^p$ and $\widetilde{\mathbf{H}}_t^v$, and their concatenation by $\mathbf{H}_t = [\widetilde{\mathbf{H}}_t^p; \widetilde{\mathbf{H}}_t^v]$.

Hierarchical evidence memory. For each frame, the fixed-size SET Tokens $\mathbf{Z}_t = \mathrm{SET}(\mathbf{H}_t)$ are stored together with dense features $\mathbf{H}_t$. Temporal grounding searches the compact tokens; dense values and inverse mappings are fetched only for frames selected for mask prediction. This hierarchical memory couples efficient temporal search with point-level reconstruction.

3.4 Find When: Searchable Evidence Tokenization

SET converts each dense observation into compact, query-independent tokens for online temporal grounding. Before compression, each Fused 3D Token is augmented with coordinate and modality embeddings so that the memory retains both spatial layout and evidence provenance without depending on a future query:

$$\begin{aligned} \mathbf{u}_{t,n} &= \mathbf{H}_{t,n} + f_{\mathrm{pos}}(\tanh(\mathbf{x}_{t,n}/s)) + \mathbf{e}_{d_{t,n}}, \\ \mathbf{Z}_t^0 &= \mathbf{E}, \qquad \mathbf{Z}_t^l = \mathrm{CrossAttn}(\mathbf{Z}_t^{l-1}, \mathbf{U}_t, \mathbf{U}_t), \\ \mathbf{Z}_t &= \mathrm{LN}(\mathbf{Z}_t^L) \in \mathbb{R}^{K\times D}. \end{aligned} \tag{2}$$

Here $\mathbf{x}_{t,n}$ is the point coordinate or voxel center, s is its normalization scale, $d_{t,n}$ is the point/voxel token type, and $\mathbf{U}_t$ stacks the augmented tokens. $\mathbf{E}$ contains $K = 32$ learnable SET slots; L and D are the cross-attention depth and token dimension. Stacked cross-attention compresses the variable-size scene into a fixed-capacity evidence set, giving every frame the same searchable budget regardless of its point count. Different learned slots can retain complementary object, attribute, and contextual evidence that a later query may require. After query arrival, token-wise language retrieval in the Activation Head compares each projected Text Token $\mathbf{T}_{q,l}$ with all SET Tokens $\mathbf{Z}_{t,k}$. Let $c_{t,l,k}^q$ denote their projected cosine similarity. Evidence is aggregated over all SET slots and then across Text Tokens to produce a raw presence logit:

$$\begin{aligned} e_{t,l}^q &= \tau_s \log\left[\frac{1}{K}\sum_{k=1}^{K}\exp(c_{t,l,k}^q/\tau_s)\right], \\ r_t^q &= \gamma_{\mathrm{set}}\sum_{l=1}^{L_q}\mathrm{softmax}_l\big(g(\mathbf{T}_{q,l})\big)e_{t,l}^q + b, \qquad \hat{a}_t^q = \mathbb{I}[r_t^q \geq \delta]. \end{aligned} \tag{3}$$

Here g is a learned Text-Token gate, γ_{set} and b are a learned scale and bias, and δ is selected on the validation set and fixed for testing. Negative frames skip mask decoding and return an empty mask. On target-present frames, the dense teacher and SET student evidence are $\mathbf{d}_{t,l}^{T,q} = \sum_n \mathrm{softmax}_n(\tilde{c}_{t,l,n}^q/\tau_d)\mathbf{H}_{t,n}$ and $\mathbf{d}_{t,l}^{S,q} = \sum_k \mathrm{softmax}_k(c_{t,l,k}^q/\tau_s)\mathbf{Z}_{t,k}$, where $\tilde{c}$ compares text with query-aligned dense tokens. $\mathcal{L}_{\mathrm{KD}} = \mathrm{mean}_{\mathcal{V}_+}[1 - \cos(\mathbf{d}^S, \mathrm{sg}(\mathbf{d}^T))]$ averages over valid Text Tokens on positive frames, with stop-gradient on the teacher. For T observed frames, N dense tokens per frame, and L_q Text Tokens, dense historical matching costs $\mathcal{O}(TL_qND)$, whereas SET retrieval costs $\mathcal{O}(TL_qKD)$ with $K \ll N$.

3.5 Find Where: Query-Conditioned Spatial Grounding

For each SET-activated frame, the spatial branch forms target-specific Q-C3D Tokens and an Object Query (Obj-Q) in three steps. First, *visual–text cross-attention* lets point and voxel tokens attend to the projected Text Tokens $\mathbf{T}_q$ through a gated residual, yielding $\mathbf{C}_t^q = [\mathbf{C}_t^{p,q}; \mathbf{C}_t^{v,q}]$. Second, *instance evidence retrieval* normalizes text–point similarity over the scene and pools the most compatible point tokens into $\mathbf{e}_t^q$, focusing the response on one referred instance. Third, *target query formation* fuses a learned scene summary, pooled language, and $\mathbf{e}_t^q$ into Obj-Q $\mathbf{o}_t^q$. The Q-C3D Tokens retain spatial context, while Obj-Q drives target decoding.

To propagate the target across activated frames, TrackState maintains a causal semantic memory and geometric state $\mathcal{T}_t^q = (\mathbf{S}_t^q, \mathbf{G}_t^q)$. Here $\mathbf{S}_t^q \in \mathbb{R}^{R\times D}$ contains $R = 4$ memory tokens, while $\mathbf{G}_t^q = (\mathbf{c}_t, \mathbf{v}_t, \mathbf{d}_t, \vartheta_t, \eta_t)$ stores center, velocity, extent, orientation, and confidence. Coordinates are mapped by the provided dataset pose into the configured global point frame. Matching the previous $\mathbf{S}_{t-1}^q$ with current Q-C3D Tokens yields a semantic bias $\mathbf{b}_t^{\mathrm{sem}}$. A constant-velocity center prediction

and normalized relative geometry yield a learned bias $\mathbf{b}_t^{\text{geo}}$. The current Obj-Q initializes the Mask Decoder, and both biases modulate its cross-attention:

$$\begin{aligned} \mathbf{q}_t &= \mathcal{D}(\mathbf{o}_t^q, \mathbf{C}_t^q; \mathbf{b}_t^{\text{sem}}, \mathbf{b}_t^{\text{geo}}), \\ \widehat{\mathbf{p}}_t^q &= \sigma(\text{Mask}(\mathbf{q}_t, \mathbf{C}_t^{p,q})) \,. \end{aligned} \tag{4}$$

The decoded object token produces point-wise mask probabilities, which are mapped back to the original cloud and binarized as $\widehat{\mathbf{m}}_t^q = \mathbb{I}[\widehat{\mathbf{p}}_t^q \geq \delta_m]$ with a fixed mask threshold δ_m. The decoded token and learned offsets cross-attend to current point tokens to form a candidate semantic memory. It initializes $\mathbf{S}_t^q$ at first activation or is fused with the previous memory by a residual gate conditioned on mask confidence and foreground margin. Reliable evidence refreshes $\mathbf{G}_t^q$; otherwise, its center is propagated over the frame interval using constant velocity while confidence decays. Historical activations follow their stored stream order; frame differences account for skipped intervals. Online SET-negative frames propagate the retained state and reset it after three consecutive negatives.

3.6 Stage-Wise Learning

QueryAnytime learns query-to-instance grounding, recurrent propagation, and temporal activation in dependency order, initializing each stage from the preceding checkpoint (Fig. 4).

The three training objectives are

$$\begin{aligned} \mathcal{L}_{\text{I}} &= \mathcal{L}_{\text{mask}} + \lambda_a \mathcal{L}_{\text{area}} + \lambda_p \mathcal{L}_{\text{p2p}} + \lambda_l \mathcal{L}_{\text{align}}, \\ \mathcal{L}_{\text{II}} &= \mathcal{L}_{\text{mask}} + \lambda_a \mathcal{L}_{\text{area}} + \lambda_p \mathcal{L}_{\text{p2p}}, \\ \mathcal{L}_{\text{III}} &= \mathcal{L}_{\text{set}} + \lambda_k \mathcal{L}_{\text{KD}}. \end{aligned} \tag{5}$$

Here $\mathcal{L}_{\text{mask}}$ combines point-wise BCE and Dice losses, $\mathcal{L}_{\text{align}}$ separates the referred instance from competing instance prototypes, $\mathcal{L}_{\text{set}}$ supervises frame-level target presence, and $\mathcal{L}_{\text{KD}}$ distills dense grounding evidence into SET Tokens.

Stage I: single-frame grounding. The spatial branch is trained on individual frames with reliable target projections using mask, area, P2P, and instance-alignment objectives.

Stage II: recurrent TrackState learning. Starting from Stage I, the model performs predicted Semantic and Geometry State rollouts over temporally consecutive observations of the same physical instance. Mask, area, and P2P losses jointly train state-conditioned decoding and prediction-driven state updates.

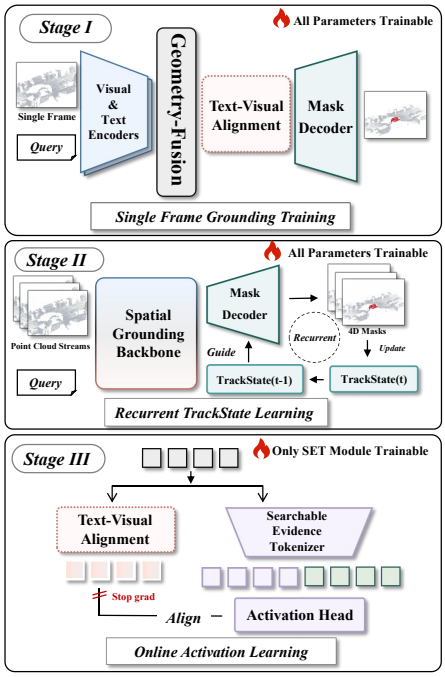


Figure 4: **Three-stage learning schedule.** Stages I–III learn spatial grounding, TrackState propagation, and SET activation. Flames mark trainable modules.

Stage III: online activation learning. Starting from Stage II, the spatial and recurrent branches are frozen. SET and the Activation Head are optimized with positive–negative retrieval supervision and Q-C3D-to-SET evidence distillation.

3.7 Online4DKITTI Construction

We construct **Online4DKITTI**, the first benchmark for online 4D query-driven point-cloud segmentation, from SemanticKITTI sequences 00–10 (Behley et al., 2019), with further construction details provided in Appendix A.2. It contains 150K query–instance–frame 3D mask records paired with fine-grained language descriptions. Each query combines category, appearance, visual relations, ego-relative position, and orientation. The **Qwen3.6-27B** vision-language model extracts open-vocabulary appearance and visual relations from quality-filtered crops and boxed scene images, while LiDAR geometry provides spatial and orientation cues. Reliability filtering composes

these cues into instance-specific descriptions, and a final VLM pass enriches queries that remain ambiguous among same-category instances. Every query–instance pair is then individually reviewed by human annotators for referential uniqueness and mask consistency.

We use sequences 00–06 for training, sequence 07 for validation, and sequences 08–10 for testing. To model realistic online interaction, we sample query arrival from an onset-centered Beta distribution over the complete sequence. Let s, b, e, and T denote the sequence start, the target's first and last observations, and the sequence end, respectively. With $m = (b - s)/(T - s)$ and concentration $\kappa = 8$, we sample

$$u^{(r)} \overset{\text{i.i.d.}}{\sim} \mathrm{Beta}(1 + \kappa m, 1 + \kappa(1 - m)), \qquad \tau_q^{(r)} = s + \min\Big(\Big\lfloor u^{(r)}(T - s + 1)\Big\rfloor, T - s\Big). \quad (6)$$

This distribution peaks at the target's first observation while retaining nonzero probability over every frame of the complete sequence. Queries may therefore arrive before the target appears, while it is present, or after it has disappeared. For each query–instance pair, the three samples define three online episodes sharing the same sequence, query, and ground-truth masks but differing in query arrival time.

4 Experiments

4.1 Experimental Setup

Evaluation metrics. Following the online protocol in Section 3.7, **Spatio-Temporal Intersection over Union (ST-IoU)** is the global IoU between predicted and ground-truth 4D masks over an episode; **Temporal Intersection over Union (T-IoU)** is the IoU between predicted and ground-truth target-presence frame sets; **Spatial Intersection over Union (S-IoU)** is the mean per-frame mask IoU over target-present frames; and **Instance Identity F1 Score (I-F1)** is the mean point-level F1 between predicted and ground-truth queried-instance masks over decoded target-present frames. Thus, target-absent frames are excluded from both S-IoU and I-F1. A missed target-present frame contributes zero to S-IoU but is excluded from I-F1 if it does not enter mask decoding; I-F1 is therefore interpreted jointly with T-IoU and S-IoU.

We evaluate online efficiency using three latency metrics. **Historical Search Latency (Search-Lat)** measures the synchronized GPU response time from query arrival until predictions for all previously observed frames are returned. **Spatial Decoding Latency (SD-Lat)** measures the average time required to process one incoming target-present frame and produce its 3D mask. **End-to-End Episode Latency (Episode-Lat)** measures the total GPU computation from query arrival until the sequence ends, including historical response and processing of every subsequent frame. All methods are evaluated under the same streaming protocol, query arrivals, hardware, numerical precision, batch size, and warm-up procedure. We report synchronized GPU compute time and exclude sensor waiting, CPU data loading, and disk I/O. Detailed latency accounting for each baseline is provided in Appendix B.

Baselines. We compare three baseline paradigms. The 3D query-driven methods are LESS, Ref-Mask3D, and 3D-STMN (Liu et al., 2024; He & Ding, 2024; Wu et al., 2024). The 4D query-driven baseline is MORE-STEM (Li et al., 2026). The language-agnostic 4D tracking baseline is Mask4D+Query Matcher, which tracks instances before matching them to the query with frozen CLIP similarity (Marcuzzi et al., 2023).

All baselines follow their original training recipes on sequences 00–06, use sequence 07 for validation, and are evaluated on test sequences 08–10. They receive identical streams and query arrivals.

Naive temporal classification. To separate spatial segmentation ability from target-presence reasoning, we retain each baseline's original result and additionally equip it with a Naive Temporal Classifier (NTC). For each frozen baseline B, NTC applies a linear logistic regressor to the final pre-decoding representation $\mathbf{h}_t^B$ selected or aggregated by its native inference rule. A positive decision invokes the native mask decoder, whereas a negative decision skips mask decoding and returns an empty mask. Only the regressor is optimized. For Mask4D, NTC instead gates whether the selected cached track mask is returned; the tracker itself remains unchanged.

Table 1: **Main comparison on Online4DKITTI.** Accuracy is in %; Search-/Episode-Lat in s/query and SD-Lat in ms/frame. Each baseline is reported with its original inference rule and an NTC variant. Latencies are rounded to whole units, with subsecond Search-Lat shown as < 1; rankings use unrounded measurements. **Bold**/underlined: best/second-best.

	Online accuracy (%)				Online efficiency		
Method	ST-IoU ↑	T-IoU ↑	S-IoU ↑	I-F1 ↑	Search-Lat ↓	SD-Lat ↓	Episode-Lat ↓
LESS	0.11	3.32	8.28	10.46	57	**35**	114
+ *NTC*	0.08	3.22	8.12	11.0	52	**35**	**103**
RefMask3D	0.23	3.32	10.32	13.71	77	47	154
+ *NTC*	0.44	7.19	9.58	12.5	71	47	141
3D-STMN	0.25	3.36	12.50	17.44	334	204	662
+ *NTC*	0.31	4.75	12.50	18.3	304	204	602
MORE-STEM	0.66	3.32	20.22	25.69	309	210	615
+ *NTC*	1.09	10.58	19.13	23.5	288	210	584
Mask4D + Query Matcher†	0.38	3.32	17.20	18.52	< 1	376	604
+ *NTC*	0.89	11.73	15.25	20.7	< 1	376	604
QueryAnytime (Ours)	**12.70**	**52.30**	**25.39**	**37.1**	1	141	212

Original LESS, RefMask3D, and MORE-STEM produce nonempty masks in every frame under their native point threshold of 0.5.
† Language-agnostic 4D panoptic tracking followed by frozen CLIP query-to-track matching with a raw similarity threshold of 0.1. Pre-query tracking is excluded from response-time latency.

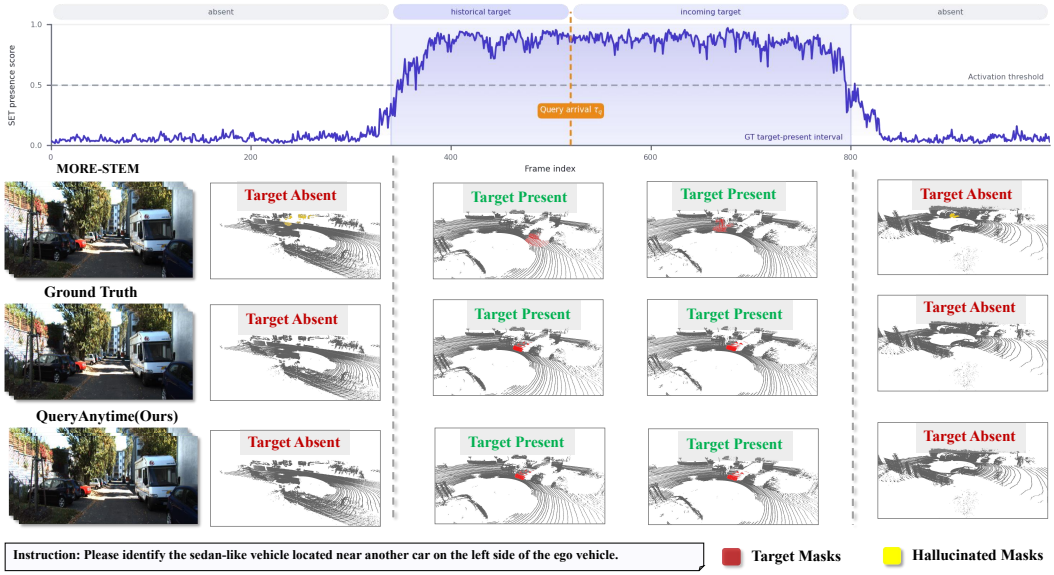


Figure 5: **Qualitative results on Online4DKITTI.** The top curve shows SET target-presence scores, with query arrival marked in orange. Red and yellow denote target and hallucinated masks, respectively. QueryAnytime suppresses absent-frame hallucinations while preserving target-present segmentation.

4.2 Main Results

Online accuracy. As shown in Table 1, QueryAnytime achieves the best result on all four accuracy metrics. It obtains 12.70 ST-IoU and 52.30 T-IoU, exceeding the strongest baselines by 11.61 and 40.57 points, respectively. SET first retrieves target-present frames, converting stronger temporal grounding into higher episode-level ST-IoU. S-IoU also rises from 20.22 to 25.39 and I-F1 from 25.69 to 37.13, demonstrating stronger spatial segmentation on decoded target-present frames; the contribution of TrackState is isolated in Table 2. Figure 5 further illustrates how temporal grounding suppresses absent-frame hallucinations while preserving target-present masks.

NTC improves T-IoU and ST-IoU for four of the five baselines: the strongest baseline results rise from 3.36 to 11.73 T-IoU and from 0.66 to 1.09 ST-IoU, whereas LESS changes from 3.32 to 3.22 T-IoU and from 0.11 to 0.08 ST-IoU. Because NTC freezes the spatial decoder and only gates which frames enter mask decoding, S-IoU stays unchanged or decreases, while conditional I-F1 also reflects the selected decoded-frame subset. Even after this adaptation, the best baseline remains well below QueryAnytime on temporal and episode-level accuracy.

Table 2: **Ablation of the principal QueryAnytime components.** Removing SET drops compact evidence and, with it, the ability to ground a query in time; removing TrackState drops recurrent target state.

	Online accuracy (%)				Online efficiency
Variant	ST-IoU ↑	T-IoU ↑	S-IoU ↑	I-F1 ↑	Episode-Lat ↓
w/o SET & TrackState	0.46	3.32	23.79	33.22	267.22
w/o SET	0.52	3.32	24.96	35.70	269.12
w/o TrackState	11.15	52.30	24.27	34.33	211.64
QueryAnytime (full)	12.70	52.30	25.39	37.13	211.70

Table 3: **Ablation of the stage-wise training strategy.** Sequential stages are compared against joint optimization.

	Online accuracy (%)			
Training strategy	ST-IoU ↑	T-IoU ↑	S-IoU ↑	I-F1 ↑
Joint training	10.54	49.91	20.22	35.67
Full sequential	12.70	52.30	25.39	37.13

Online efficiency. QueryAnytime records 1 s Search-Lat and 212 s Episode-Lat. Across the NTC variants, Search-Lat ranges from below 1 to 304 s, while Episode-Lat ranges from 103 to 604 s. For query-conditioned baselines, NTC reduces Search-Lat and Episode-Lat by skipping final mask decoding on negative frames. Because positive frames use the unchanged native decoder, SD-Lat remains unchanged at the reported whole-unit precision for every baseline pair. Mask4D+Query Matcher instead benefits from cached pre-query tracks, while LESS and RefMask3D retain lower per-frame latency. Considering accuracy and latency together, QueryAnytime offers a favorable accuracy–efficiency trade-off.

4.3 Ablation Studies

Overall architecture. Table 2 separates the contributions of temporal evidence retrieval and recurrent propagation. Removing SET reduces T-IoU from 52.30 to 3.32 and ST-IoU from 12.70 to 0.52, while increasing Episode-Lat from 212 s to 269 s; S-IoU changes only from 25.39 to 24.96. SET therefore primarily determines *when* to decode. Removing TrackState leaves T-IoU unchanged and latency nearly unchanged, but lowers S-IoU, I-F1, and ST-IoU by 1.12, 2.80, and 1.55 points, respectively, confirming its complementary role in determining *where* and how accurately to segment the target.

Stage-wise learning. As reported in Table 3, the full sequential strategy improves ST-IoU, T-IoU, S-IoU, and I-F1 by 2.16, 2.39, 5.17, and 1.46 points, respectively. The largest gain in S-IoU supports learning spatial grounding, TrackState propagation, and SET distillation in dependency order rather than optimizing all objectives jointly.

5 Conclusion

In this work, we studied Online Query-driven 4D Point Cloud Segmentation, where a language query may arrive at an arbitrary time while 3D observations continue to evolve. In this setting, the model must dynamically identify the query-relevant temporal span and segment the referred instance within it. We summarized this challenge as "*when* before *where*" and proposed **QueryAnytime**. For *when*, SET builds a compact, query-independent evidence space for efficient temporal grounding without revisiting dense historical observations. For *where*, TrackState maintains a structured semantic-geometric target state for instance-consistent segmentation across relevant observations. We further introduced **Online4DKITTI**, a dedicated benchmark that decouples query arrival time from target visibility. Experiments demonstrate that QueryAnytime consistently outperforms existing baselines in segmentation accuracy while maintaining acceptable end-to-end latency.

AI USE STATEMENT

In this work, generative AI tools were used to assist with the construction of Online4DKITTI by generating language annotations, including appearance and static spatial descriptions, resolving same-category ambiguities, and verifying query grounding, as detailed in Appendix A.2. Beyond dataset construction, generative AI tools were not used for research ideation, methodology or experimental design, method implementation, mathematical claims or proofs, result interpretation, or any other research task. AI-generated annotations were filtered and verified using the fixed pipeline and prompts described in Appendix A.2, We take responsibility for the final content of this work.

REFERENCES

Panos Achlioptas, Ahmed Abdelreheem, Fei Xia, Mohamed Elhoseiny, and Leonidas J. Guibas. ReferIt3D: Neural listeners for fine-grained 3d object identification in real-world scenes. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 422–440, 2020. doi: 10.1007/978-3-030-58452-8_25.

Mehmet Aygun, Aljosa Osep, Mark Weber, Maxim Maximov, Cyrill Stachniss, Jens Behley, and Laura Leal-Taixe. 4d panoptic LiDAR segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5527–5537, June 2021.

Shehreen Azad, Vibhav Vineet, and Yogesh S. Rawat. Streamready: Learning what to answer and when in long streaming videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 40494–40504, 2026.

Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Juergen Gall. SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9296–9306, 2019.

Dave Zhenyu Chen, Angel X. Chang, and Matthias Nießner. ScanRefer: 3d object localization in RGB-D scans using natural language. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 202–221, 2020. doi: 10.1007/978-3-030-58565-5_13.

Shuting He and Henghui Ding. RefMask3D: Language-guided transformer for 3d referring segmentation. In *Proceedings of the ACM International Conference on Multimedia (ACM MM)*, pp. 8316–8325, 2024. doi: 10.1145/3664647.3680998.

Shuting He, Henghui Ding, Xudong Jiang, and Bihan Wen. SegPoint: Segment any point cloud via large language model. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 349–367, 2024. doi: 10.1007/978-3-031-72670-5_20.

Pin-Hao Huang, Han-Hung Lee, Hwann-Tzong Chen, and Tyng-Luh Liu. Text-guided graph neural networks for referring 3d instance segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 1610–1618, 2021. doi: 10.1609/aaai.v35i2.16253.

Zhenpeng Huang, Xinhao Li, Jiaqi Li, Jing Wang, Xiangyu Zeng, Cheng Liang, Tao Wu, Xi Chen, Liang Li, and Limin Wang. Online video understanding: OVBench and videochat-online. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3328–3338, 2025.

Minsoo Kim, Kyuhong Shim, Jungwook Choi, and Simyung Chang. InfiniPot-V: Memory-constrained KV cache compression for streaming video understanding. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Chade Li, Haida Feng, Pengju Zhang, and Yihong Wu. MORE-STEM: Long-short MemOry REcall and Spatio-TEmporal consistency model for query-driven 3d/4d point cloud segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 31462–31471, June 2026.

Xuexun Liu, Xiaoxu Xu, Jinlong Li, Qiudan Zhang, Xu Wang, Nicu Sebe, and Lin Ma. LESS: Label-efficient and single-stage referring 3D segmentation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024.

Rodrigo Marcuzzi, Lucas Nunes, Louis Wiesmann, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. Contrastive instance association for 4d panoptic segmentation using sequences of 3d LiDAR scans. *IEEE Robotics and Automation Letters*, 7(2):738–745, 2022. doi: 10.1109/LRA.2022.3140439.

Rodrigo Marcuzzi, Lucas Nunes, Louis Wiesmann, Elias Marks, Jens Behley, and Cyrill Stachniss. Mask4D: End-to-end mask-based 4d panoptic segmentation for LiDAR sequences. *IEEE Robotics and Automation Letters*, 8(11):7487–7494, 2023. doi: 10.1109/LRA.2023.3320020.

Junbo Niu, Yifei Li, Ziyang Miao, Chunjiang Ge, Yuanhang Zhou, Qihao He, Xiaoyi Dong, Haodong Duan, Shuangrui Ding, Rui Qian, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, and Jiaqi Wang. OVO-Bench: How far is your video-LLMs from real-world online video understanding? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18902–18913, 2025.

Changli Wu, Yiwei Ma, Qi Chen, Haowei Wang, Gen Luo, Jiayi Ji, and Xiaoshuai Sun. 3D-STMN: Dependency-driven superpoint-text matching network for end-to-end 3d referring expression segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 5940–5948, 2024. doi: 10.1609/aaai.v38i6.28408.

Dongming Wu, Tiancai Wang, Yuang Zhang, Xiangyu Zhang, and Jianbing Shen. Onlinerefer: A simple online baseline for referring video object segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2761–2770, 2023.

Jiannan Wu, Yi Jiang, Peize Sun, Zehuan Yuan, and Ping Luo. Language as queries for referring video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4974–4984, 2022.

Yanlai Yang, Zhuokai Zhao, Satya Narayan Shukla, Aashu Singh, Shlok Kumar Mishra, Lizhu Zhang, and Mengye Ren. StreamMem: Query-agnostic KV cache memory for streaming video understanding. *arXiv preprint arXiv:2508.15717*, 2025.

Supplementary Materials

A Online4DKITTI Benchmark Details

A.1 Benchmark Overview

Online4DKITTI is constructed from SemanticKITTI sequences 00–10 and fine-grained language annotations. We distinguish four data units to make the reported scale unambiguous. A **physical instance** is one temporally associated SemanticKITTI instance track, uniquely identified by its sequence and instance IDs. A **query–instance pair** associates one natural-language description with its target physical instance; one instance may have multiple queries. A **frame-level 3D mask record** is a query–instance–frame annotation. An **online episode** combines one query–instance pair with one sampled query-arrival position.

We follow the SemanticKITTI sequence split: 00–06 for training, 07 for validation, and 08–10 for testing. Table 4 summarizes the consolidated benchmark counts. Three arrival samples per query–instance pair produce 4,431 online episodes;

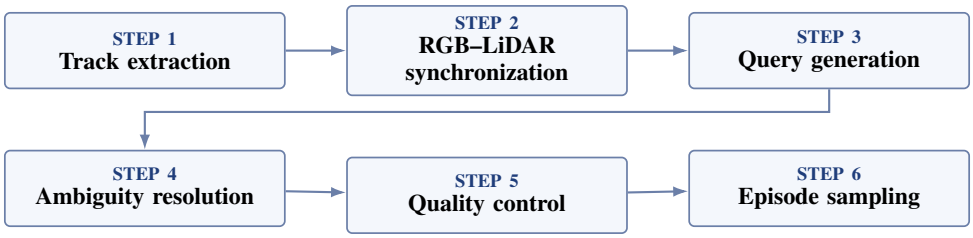
Table 4: **Online4DKITTI scale by data split.**

		Dataset units		Online evaluation	
Split	**Seq.**	**Pairs**	**Mask records**	**Episodes**	**Frames/query**
Train	00–06	1,165	91,799	3,495	78.797
Val	07	139	13,407	417	96.453
Test*	08–10	173	45,177	519	462.688
Total	**00–10**	**1,477**	**150,383**	**4,431**	**125.424**

* Episodes use three sampled arrivals per query–instance pair and share the same masks.

A.2 Benchmark Construction Pipeline

Online4DKITTI is constructed from the LiDAR scans, RGB images, camera calibration, ego poses, and panoptic instance labels provided by SemanticKITTI. The pipeline consists of instance-track extraction, multimodal synchronization, query generation, ambiguity resolution, quality control, and online-episode sampling.



VLM and fixed settings. All VLM-based annotation steps use **Qwen3.6-27B**. Appearance generation receives the complete crop sequence of the target; static-relation generation receives the complete sequence of full-scene images with the target boxed in red; peer disambiguation receives the complete grouped sequences of all same-category candidates. The final grounding-verification pass likewise uses the complete target sequence. Here, "complete" refers to all available frames retained after the visual-quality filtering described below. We use the exact prompts below and constrain each call to a short phrase or an explicit failure token (`INVALID` or `IMPOSSIBLE`). Episode construction uses three arrival samples per query–instance pair, the onset-centered Beta distribution in Eq. 6 with concentration $\kappa = 8$, and a fixed random seed.

Instance-track extraction. We first decompose each SemanticKITTI panoptic label into its semantic and instance IDs. Observations with the same `(semantic_id, instance_id)` in a sequence are temporally ordered to form one physical instance track. Each track frame stores the

target LiDAR points, point-level instance mask, frame index, center, point count, projected region, ego pose, and visibility metadata. Short occlusion gaps are merged when the observations on both sides belong to the same continuous instance. Tracks with too few observations or without stable temporal support are removed. Semantic classes are obtained from the SemanticKITTI label mapping; ignored classes and categories unsuitable for query generation are filtered out.

RGB–LiDAR synchronization and target-region generation. Using the official camera calibration and ego poses, we project target LiDAR points into the synchronized RGB image. The projected points define a tight target box and RGB crop; we also retain the full-scene image with the target box overlaid. Target crops support appearance annotation, whereas boxed scene images support static spatial-relation annotation. Images are rejected for visual-attribute generation when the projection contains too few target points, the target is too small, severely occluded, excessively dark, or strongly blurred.

Appearance generation. The complete sequence of retained target crops is passed to Qwen3.6-27B. The model may describe only visually supported evidence and is explicitly prohibited from inferring license plates, brands, exact vehicle models, or invisible details.

Appearance prompt

```
You are given a complete sequence of cropped images of the same
target instance.  The target category is "{category}".  Describe the
target's visible appearance when any trustworthy visual evidence
is available.  Prefer a coarse but reliable phrase over INVALID:
color, broad subtype, clothing, rider, cargo, lights, markings, or
shape may be used only when visible.  Output INVALID only when the
target itself cannot be reliably identified in the crop sequence
because it is extremely tiny, almost fully outside the crops, too
dark or blurred to recognize, severely occluded, or confused with
another object.  Do not invent fine details such as license plates,
logos, text, exact model, or accessories unless they are clearly
readable.  Output one short English noun phrase describing only
visible appearance, or exactly INVALID. Do not mention position,
image order, scene context, crop quality, or uncertainty.  Do not
explain.  Examples:  "a white sedan", "a dark car", "a cyclist in
dark clothing", "a blue truck with a white cab".
```

If the model returns `INVALID`, the appearance field is omitted and the query falls back to the remaining reliable dimensions.

Static 2D-relation generation. The complete sequence of retained full-scene images with a red target box is passed to Qwen3.6-27B. The output is restricted to static spatial descriptions.

Static spatial-relation prompt

```
You are given a complete sequence of full driving-scene images.  The
target object is marked with a red bounding box in every retained
frame.  The target category is "{category}".  Describe a reliable
visual spatial relation for the marked target when the target
and at least one visible reference are clear enough.  Prefer a
specific static visual relation to broad left/right wording when a
trustworthy reference exists.  Use lanes, curb, road boundary, nearby
vehicles, pedestrians, cyclists, buildings, barriers, vegetation,
or other stable visible scene elements.  Output INVALID only when
the marked target is not recognizable, the box is too tiny or
unclear, the target is severely occluded, or no reliable visual
reference is visible.  Output one short English phrase describing
only spatial layout, or exactly INVALID. Use only static spatial
relations; do not use terms such as parked, stopped, stationary,
moving, driving, traveling, turning, waiting, approaching, leaving,
crossing, following, or overtaking.  Do not mention the red box,
```

Static spatial-relation prompt (continued)

```
image order, exact pixel position, camera, or uncertainty. Do not
explain. Examples: "near the curb", "beside a white truck", "behind
a silver car", "between two cars on the right side", "next to the
road barrier".
```

3D relation and orientation. Geometric relations are computed from the target LiDAR points and instance track. Robust cross-frame statistics estimate ego-relative distance and stable 3D location; a relation is retained only when it has sufficient and consistent support across frames. Orientation is estimated from the principal direction of the target points on the ground plane and compared with the ego direction. Unstable orientation estimates are omitted rather than converted into uncertain language.

Same-category ambiguity resolution. When multiple instances of the same category receive identical or near- identical queries, they are grouped as peer candidates. Qwen3.6-27B is then given the complete retained image sequence of every candidate and asked to find a visible appearance cue that distinguishes each target from all other peers.

Peer-disambiguation prompt

```
The image groups are peer candidate instances that currently share
the same text query. No candidate is preferred, primary, default,
or a reference for the others. Candidate {i}: images {ranges}.
Describe Candidate {i}. Return one short English noun phrase that
distinguishes this candidate from every other candidate group. Use
only visible appearance such as color, subtype, clothing, rider,
cargo, lights, text, markings, or shape. Do not mention position,
image order, image numbers, or the scene. If no distinctive visible
attribute is clear, output exactly IMPOSSIBLE. Output only the
phrase, for example: "a silver sedan with dark windows" or "a
cyclist in a red jacket".
```

If no reliable distinguishing cue exists, the sample is marked as unresolved rather than being assigned an artificial distinction.

Query verification and online-episode construction. After generation, each query is checked for unique target grounding and for mask consistency over its physical instance track. The final VLM verification pass also uses Qwen3.6-27B together with the complete retained target sequence and the following prompt:

Grounding-verification prompt

```
Confirm the query grounding for "{query}". Answer with a sentence
containing [SEG].
```

After quality control, each query–instance pair is converted into three online episodes. Arrival times are sampled anywhere in the complete sequence using the fixed onset-centered Beta distribution with $\kappa = 8$ and a fixed random seed. Figure 6 illustrates the query-arrival support and target-visible interval on their shared episode timeline. Each episode retains both the historical prefix and all subsequent online frames. Official test episodes use sequences 08–10, with three distinct arrivals generated whenever possible. The final manifest stores the query, target instance, arrival time, historical and future frames, target-presence labels, and corresponding 3D masks so that every method receives exactly the same online inputs and evaluation split.

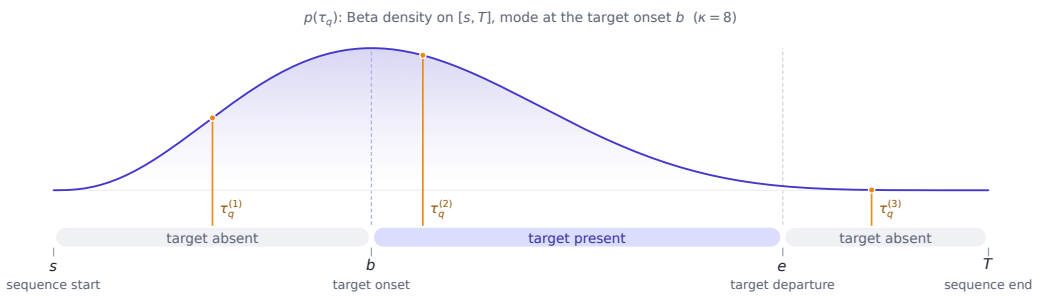


Figure 6: **Online query-arrival sampling and target visibility.** All elements share the same normalized time axis. The onset-centered Beta distribution in Eq. 6 is supported on $[s, T]$ and peaks at the target onset b for $\kappa = 8$. The target is present only during $[b, e]$ and absent elsewhere in the sequence. The three markers show example arrivals before, during, and after target presence. The timeline is schematic rather than a dataset statistic.

B Accuracy, Latency, and Baseline Adaptation

B.1 Accuracy Metrics

For an episode of T frames, let $\mathbf{m}_t$ and $\hat{\mathbf{m}}_t$ denote the ground-truth and predicted point masks, and let $a_t = \mathbb{I}[\|\mathbf{m}_t\|_1 > 0]$ and $\hat{a}_t = \mathbb{I}[\|\hat{\mathbf{m}}_t\|_1 > 0]$. We compute

$$\begin{aligned} \text{ST-IoU} &= \frac{\sum_{t=1}^{T} |\hat{\mathbf{m}}_t \cap \mathbf{m}_t|}{\sum_{t=1}^{T} |\hat{\mathbf{m}}_t \cup \mathbf{m}_t|}, \qquad \text{T-IoU} = \frac{\sum_{t=1}^{T} \hat{a}_t a_t}{\sum_{t=1}^{T} \mathbb{I}[\hat{a}_t + a_t > 0]}, \\ \text{S-IoU} &= \frac{1}{|\mathcal{P}|} \sum_{t \in \mathcal{P}} \frac{|\hat{\mathbf{m}}_t \cap \mathbf{m}_t|}{|\hat{\mathbf{m}}_t \cup \mathbf{m}_t|}, \qquad \text{I-F1} = \frac{1}{|\mathcal{D}|} \sum_{t \in \mathcal{D}} \frac{2\text{TP}_t}{2\text{TP}_t + \text{FP}_t + \text{FN}_t}, \end{aligned} \tag{7}$$

where $\mathcal{P} = \{t : a_t = 1\}$ contains target-present frames and $\mathcal{D} \subseteq \mathcal{P}$ contains those entering mask decoding. A missed target-present frame contributes zero to S-IoU but is not included in I-F1.

B.2 Latency Protocol

All methods use identical hardware, numerical precision, batch size, warm-up, and GPU synchronization. Sensor waiting, CPU loading, and disk I/O are excluded. Let τ_q be the query-arrival frame, $\mathcal{H} = \{1, \ldots, \tau_q\}$, and $\mathcal{F} = \{\tau_q + 1, \ldots, T\}$. Search-Lat is strictly the synchronized GPU response time from query arrival until predictions for $\mathcal{H}$ are returned. If c_t is the measured GPU cost for an incoming frame and $\mathcal{D}_{\text{fut}} \subseteq \mathcal{F}$ denotes decoded target-present frames, then

$$L_{\text{SD}} = \frac{1}{|\mathcal{D}_{\text{fut}}|} \sum_{t \in \mathcal{D}_{\text{fut}}} c_t, \qquad L_{\text{episode}} = L_{\text{search}} + \sum_{t \in \mathcal{F}} c_t. \tag{8}$$

The second sum includes every future frame, including target-absent frames for which no mask is produced.

B.3 Baseline Adaptation with NTC

NTC uses the final pre-decoding representation $\mathbf{h}_t^B$ exposed by each baseline's native inference path. All baseline parameters remain frozen. For the query-conditioned baselines, a positive logistic-regression decision invokes the native mask decoder, whereas a negative decision skips mask decoding and returns an empty mask. For Mask4D, the decision gates whether the selected cached track mask is returned while tracking continues unchanged. The method-specific inputs are summarized below.

Baseline	NTC input $\mathbf{h}_t^B$
LESS	Sentence-conditioned weighted sum of final queries supplied to the output mask decoder.
RefMask3D	Selected final decoder query after mask-embedding projection and before point-mask prediction.
3D-STMN	Normalized language-enhanced query selected for superpoint mask prediction.
MORE-STEM	Memory-refined query-conditioned feature passed to the final segmentation head.
Mask4D+Query Matcher	Selected track mask embedding plus its frozen-CLIP query similarity.

B.4 Method-Specific Cost Accounting

Frame-wise 3D baselines. Without NTC, LESS, RefMask3D, and 3D-STMN run their complete query-conditioned single-frame pipeline independently on every frame. Thus, $L_{\text{search}} = \sum_{t\in\mathcal{H}} c_t^{\text{full}}$, and the same full cost is incurred for every future frame.

NTC-gated decoding. For LESS, RefMask3D, 3D-STMN, and MORE-STEM, let $z_t \in \{0,1\}$ denote the NTC decision. Their per-frame cost becomes

$$c_t^{B+\text{NTC}} = c_t^{\text{pre}} + c_t^{\text{NTC}} + z_t c_t^{\text{mask}}, \tag{9}$$

where c_t^{pre} ends at $\mathbf{h}_t^B$ and c_t^{mask} is the skipped final mask-decoding cost. This gating reduces Search-Lat and Episode-Lat on NTC-negative frames, while decoded positive frames retain the baseline's mask decoder and SD-Lat.

MORE-STEM and Mask4D+Query Matcher. MORE-STEM processes the historical prefix with the query and initializes its temporal memory; its causal feature and memory updates remain active, while NTC gates the final segmentation head as in Equation 9. Mask4D instead builds language-agnostic tracks before query arrival. This pre-query computation is outside response-time latency, so its Search-Lat contains only cache access and frozen-CLIP query-to-track matching; at each frame, the highest-similarity track is selected, with no mask returned when its raw cosine similarity is below 0.1. Future frames include tracking and mask updates regardless of target presence, and Mask4D+NTC only gates whether the selected track mask is returned.

QueryAnytime. Query-independent visual encoding and SET construction completed before query arrival are likewise excluded from Search-Lat. With $\hat{a}_t^q \in \{0,1\}$ denoting SET activation, its response and future-frame costs are

$$\begin{aligned} L_{\text{search}}^{\text{QA}} &= c_{\text{text}} + \sum_{t\in\mathcal{H}} \big(c_t^{\text{SET}} + \hat{a}_t^q c_t^{\text{decode}}\big), \\ c_t^{\text{QA}} &= c_t^{\text{encode}} + c_t^{\text{SET}} + \hat{a}_t^q \big(c_t^{\text{TrackState}} + c_t^{\text{decode}}\big), \qquad t \in \mathcal{F}. \end{aligned} \tag{10}$$

SET-negative future frames therefore still contribute encoding and activation cost to Episode-Lat. SD-Lat is conditional on target-present frames entering mask decoding, so it is interpreted together with T-IoU.