Technical Report for CVPR 2026 Workshop on Argoverse 2 Scenario Mining Challenge: A Multi-Generator and VLM Solution

Ziang Wei Minjun Yu Mingjie Pang Zheyuan Lai Kewei Wang Wei Li
EABOT
{ziang.wei, gene.yu, jayden.pang, jory.lai, kewei.wang, alex}@eabot.tech

Abstract

We report on our entry to the CVPR 2026 Argoverse 2 Scenario Mining (RefAV) challenge, where a natural language prompt must be grounded to the object tracks in a driving log that satisfy it. Following the RefAV baseline, a large language model composes hand-crafted atomic functions into an executable program that filters tracker outputs into a referred-object set. Our team's best result on the temporal track is a TBA of 72.86 (LogBA of 71.09), which placed us 5th on the challenge leaderboard. Each prompt is sent to two LLM generators, a GPT-5.5 baseline and Opus 4.7 at low effort, each of which writes a small domain-specific program. We merge their outputs by union, with the baseline as the primary source. Prompts that the merge still leaves empty are rescued by rewriting the prompt and regenerating its program with a high-effort Opus 4.7 pass. A final post-processing stage refines the temporal windows and uses a vision-language model (Qwen3.6 Plus) to check each candidate timestamp against the camera frame so that the per-frame predictions match what is actually visible.

1. Introduction

Modern autonomous vehicles stream on the order of terabytes of multi-sensor data per hour, the overwhelming majority of which captures uninteresting driving. Retrieving the rare, safety-relevant scenarios (an unprotected left turn, a jaywalking pedestrian, a truck changing lanes ahead) is the bottleneck for evaluation, safety testing, and active learning at scale. The RefAV benchmark [1] formalizes this as *scenario mining*: given a natural-language prompt and a log of 3D object tracks, return the set of tracks (and the time intervals) that satisfy the prompt. RefAV is the basis of the CVPR 2026 Argoverse 2 Scenario Mining Challenge, which scores a temporal track (TBA/LogBA) and a spatio-temporal track (HOTA-Temporal). We take the temporal track, scored by Temporal Ball Accuracy (TBA), as our primary objective.

The RefAV baseline solves each prompt with code generation: a large language model (LLM) composes a small library of hand-crafted atomic functions into an executable predicate over the track set. This design is attractive because every prediction is a short, inspectable program, and because progress reduces to producing better programs for the cells that the current system gets wrong.

This report documents our challenge effort. Our team's best result on the temporal track is a TBA of 72.86 (LogBA of 71.09), produced by the pipeline of Fig. 1. This entry ranked 5th on the final challenge leaderboard (Tab. 1). Each prompt is sent to two LLM generators, a GPT-5.5 baseline and Opus 4.7 at low effort, and each writes a small domain-specific program. We merge their outputs by union, with the baseline as the primary source. Prompts that the merge still leaves empty are rescued by rewriting the prompt and regenerating its program with a high-effort Opus 4.7 pass. A final post-processing stage refines the temporal windows and uses a vision-language model (Qwen3.6 Plus) to check each candidate timestamp against the corresponding camera frame. This post-processing yields frame-accurate temporal activations and is the main driver of our high TBA (Sec. 2).

2. Method and Team Results

2.1. RefAV Pipeline: RefProg

We adopt the RefAV baseline RefProg method [1]. A 3D tracker (Le3DE2E [3]) first produces object tracks for each log. For every (log, prompt) *cell*, an LLM is prompted to emit a single self-contained program in a small domain-specific language (DSL) that composes *atomic functions* into a predicate over the track set. The program is run, exactly as generated, against the Argoverse 2 log and returns the referred-object set, i.e. the subset of tracks (and per-timestamp activations) that satisfy the prompt. Because each prompt is realized as executable code, the search space is the space of compositions of a fixed, hand-crafted function library, which makes generation cheap to validate and easy to inspect. We build on this baseline with two heterogeneous generators and an empty-cell rescue (Sec. 3), followed by the post-processing below.

2.2. Post-Processing: Temporal Windows and Visual Refinement

The DSL program yields, for each referred object, the candidate time intervals during which the prompt is predicted to hold. Because TBA is a *frame-level* metric (it rewards getting the active/inactive label right at each individual timestamp), temporal precision at interval boundaries matters as much as object identity. We sharpen these intervals in two steps.

First, we *refine* the temporal windows directly: we fill in missing frames that fall between two detections of the same object, and we expand the edges of each active interval so that short gaps and clipped boundaries left by the geometry-only predicate are smoothed over. This step is purely algorithmic and needs no model.

Second, we add a visual-refinement stage. For each candidate timestamp we pass the corresponding ring-camera frame, together with the natural language prompt, to a vision-language model (Qwen3.6 Plus), which judges whether the described scenario is actually visible in that frame. Frames the model rejects are removed from the activation and the boundaries are tightened to the frames it confirms. This per-frame visual grounding is the principal driver of our high TBA: it converts coarse, geometry-only interval predictions into frame-accurate activations, correcting boundary timestamps that the track-level predicate alone leaves imprecise.

2.3. Tracks and Metrics

The challenge comprises two tracks. The temporal track is evaluated using Temporal Ball Accuracy (TBA) and balanced log-level accuracy (LogBA), whereas the spatio-temporal track is evaluated using HOTA-Temporal (HOTA-T) and HOTA-Track [2]. We treat *TBA as the primary objective*: it is the frame-level metric used to rank submissions on the temporal track, and it is the metric most directly affected by our post-processing in Sec. 2.2. LogBA serves as a log-level guard against regressions. We report the spatio-temporal HOTA scores for completeness. We report our headline scores and the diagnostics that frame the remaining headroom in Sec. 4.

3. Combining Two Generators

3.1. Motivation

Reusing our base system's own programs no longer adds predictions for the cells it still gets wrong, either by leaving them empty or by selecting the wrong tracks (Sec. 2). For further improvement, we need a new source of programs. However, simply adding a single new model is risky: a model that is stronger on average may still underperform the base model on individual cells. Rather than replacing the base system, we augment it. Specifically, we run a second generator in parallel with the original one and merge their outputs, so that the combined system can fill empty cells or strengthen weak predictions without overwriting predictions that the base system already gets right (Fig. 1).

3.2. Two Generators

The primary generator is GPT-5.5, the primary source of the composite: fast, cheap, and correct on most cells. The second generator is Opus 4.7 at low effort. Each receives the same natural language prompt and writes its own DSL program over the atomic-function library, and both programs are run on the log, producing two referred-object sets per prompt.

3.3. Merging

We merge the two sources by taking, for each prompt, the union of their references, treating GPT-5.5 as the primary source. Because the union only ever adds references, it does not drop a correct baseline prediction. It fills prompts that one generator leaves empty and broadens prompts where the two disagree, while leaving the baseline's correct predictions untouched.

3.4. Empty-Cell Rescue with a Polished Prompt

Some prompts are still left empty after the merge, typically because both generators misread the original wording. For these we do not reuse the same prompt: we *rewrite* it into a clearer, more explicit form and regenerate its program with a high-effort Opus 4.7 pass, which is slower but stronger on compositional reasoning. The rescued programs fill the remaining empty prompts, and the merged-and-rescued referred-object sets are what we then post-process (Sec. 2.2).

4. Results

4.1. Our Best Result

Our best submission is the pipeline of Sec. 2 and Fig. 1, which achieves a TBA of 72.86 and a LogBA of 71.09 on the temporal track. This entry placed us 5th on the final challenge leaderboard (Tab. 1), within 4.4 TBA points of the top entry. The high frame-level score (TBA) is primarily driven by the post-processing of Sec. 2.2. Combining two generators and rescuing empty prompts (Sec. 3) broadens coverage before post-processing is applied.

4.2. Post-Processing the Temporal Window

The single most effective lever on TBA was engineering the *temporal window* of each activation. The DSL program returns coarse, geometry-only intervals whose boundaries are frequently a few frames too early or too late, and which occasionally span a stretch in which the scenario is not actually visible. We correct these in two steps (Sec. 2.2).

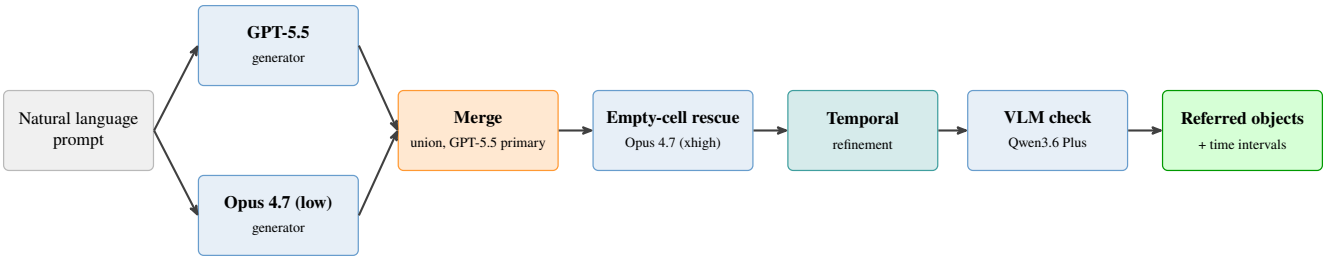

Figure 1. **Our scenario-mining pipeline.** Each natural language prompt is sent to two LLM generators, GPT-5.5 (our baseline) and Opus 4.7 at low effort; each writes a domain-specific program over hand-crafted atomic functions that is run on the tracker output. We *merge* the two referred-object sets by union, with GPT-5.5 as the primary source. Prompts that the merge still leaves empty are rescued by rewriting the prompt and regenerating its program with a high-effort Opus 4.7 (xhigh) pass. The *temporal refinement* stage then post-processes the per-frame predictions of each referred object: it is a deterministic, model-free pass that treats every track as an interval signal, filling short gaps where a brief detection dropout would otherwise split one continuous event into fragments and expanding interval edges so that the predicted window covers the full extent of the described behaviour rather than only its most confident core. This smoothing is what aligns the frame-level mask with the ground-truth interval and is a key driver of Temporal Ball Accuracy (TBA), since TBA rewards correct predictions at every timestamp, not just at the moment of peak confidence. Finally, a vision-language model (Qwen3.6 Plus) checks each candidate timestamp against the camera frame, so that the per-frame predictions match what is actually visible. Blue blocks are model-based stages; the rest are deterministic.

Rank	Team	TBA	LogBA
1	MISI (AutoMine)	77.21	76.26
2	MTL (Argonaut)	75.50	80.75
3	HYU_OASIS	74.69	77.84
4	DataClaw	74.18	78.67
5	EABOT_AMD (ours)	72.86	71.09

Table 1. Final leaderboard of the CVPR 2026 Argoverse 2 Scenario Mining (RefAV) challenge temporal track, ranked by Temporal Ball Accuracy (TBA). Our entry (EABOT_AMD) placed 5th.

- **Temporal refinement** We fill missing frames that fall between two detections of the same object and expand the edges of each active interval, smoothing over short gaps and clipped boundaries left by the geometry-only predicate. This recovers true-active timestamps that a strict geometric threshold would otherwise drop.
- **Visual refinement** For each candidate timestamp, the ring-camera frame and the prompt are passed to Qwen3.6 Plus. Frames at which the model does not see the described scenario are removed, shrinking over-extended intervals to the frames that are actually consistent with the prompt. This eliminates the false-active timestamps that most directly cost TBA.

Since TBA is scored per frame, these boundary corrections translate directly into score: temporal refinement recovers missed true positives at interval edges while visual refinement removes false positives. Together, they account for the bulk of the gap between the raw geometry-only predictions and our final submission, and they are the reason our frame-level accuracy is high relative to the track-level metrics.

5. Conclusion

We studied Argoverse 2 scenario mining in the CVPR 2026 challenge, where a natural language prompt is grounded to the object tracks in a driving log that satisfies it. Our best entry reaches a TBA of 72.86 on the temporal track. Each prompt is sent to two LLM generators, a GPT-5.5 baseline and Opus 4.7. Each generator produces a small program, and their outputs are merged via union, while retaining the baseline as the primary source. Prompts that the merge still leaves empty are rescued by rewriting the prompt and regenerating its program with an xhigh effort Opus 4.7 pass. Finally, a post-processing stage refines the temporal windows by using a vision-language model to verify each candidate timestamp against the corresponding camera frame.

References

[1] Cainan Davidson, Deva Ramanan, and Neehar Peri. Refav: Towards planning-centric scenario mining. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026. 1

[2] Jonathon Luiten, Aljosa Osep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. HOTA: A higher order metric for evaluating multi-object tracking. *International Journal of Computer Vision*, 129:548–578, 2021. 2

[3] Zhepeng Wang, Feng Chen, Kanokphan Lertniphonphan, Siwei Chen, Jinyao Bao, Pengfei Zheng, Jinbao Zhang, Kaer Huang, and Tao Zhang. Technical report for argoverse challenges on unified sensor-based detection, tracking, and forecasting. *arXiv preprint arXiv:2311.15615*, 2023. 1