Bridging Structure and Language: Graph-Based Visual Reasoning for Autonomous Road Understanding

Lena Wild[1,2] **Katie Z Luo**[3,4] **Marco Pavone**[3,5]
[1]KTH Royal Institute of Technology [2]TRATON [3]Stanford University
[4]Waymo [5]NVIDIA
lwild@kth.se, {katieluo, pavone}@stanford.edu

Abstract

Structured road understanding of lane geometry, topology, and traffic element relationships is foundational to safe autonomous driving. While vision-language models (VLMs) offer promising semantic flexibility, they lack the geometric and relational grounding required for precise road reasoning. Conversely, traditional modular systems, *e.g.*, HD maps and topological road graphs, provide structural precision but remain semantically rigid. To bridge this gap, we introduce the Combined Road Substrate (CRS), a graph-grounded framework that makes geometric road structure and open-vocabulary semantics jointly executable in a single representation. CRS enables the automatic generation of compositionally complex and linguistically varied question-answer pairs via recursive graph queries, augmented with a "grounding for free" mechanism that ensures logical traceability to specific map elements, and procedurally extracted chain-of-thought supervision traces. We demonstrate that state-of-the-art VLMs—including large, closed-source models—struggle significantly with structured road reasoning, yet training a small 2- or 4-billion-parameter model with as few as 20 to 80 CRS-enriched scenes yields stable gains in compositional reasoning tasks of varying depth. Analysis of model behavior via verifiable reasoning traces reveals a systematic shift in failure modes: whereas baseline models fail at relational scene understanding, CRS-trained models reduce failures to attribute recognition, suggesting that for structured road reasoning tasks, the lack of structured supervision is a greater limitation than model scale. Project page: https://kth-rpl.github.io/CRS/.

1 Introduction

For self-driving cars to operate safely, they must be able to reason about the roads they are on—understanding lane structure and geometry, interconnectivity, and relationships between traffic elements such as signs and lights—to determine what behaviors are permitted. This geometrical and relational "substrate" of road understanding forms the foundation for downstream decision-making, and has motivated extensive work in road map prediction [1, 2, 3], topology reasoning [4, 5], and methods for incorporating high- and standard-definition maps into downstream tasks [6]. In such pipelines, structured road understanding is explicitly represented and can be directly supervised.

More recently, however, end-to-end driving approaches [7, 8] have gained traction, learning policies that map observations directly to actions. While effective, these methods often do not rely on explicit intermediate road representations, obscuring the causal link between road structure and vehicle behavior. Similarly, foundation models such as vision-language models (VLMs) [9, 10] exhibit strong generalist capabilities, but often rely on coarse visual and semantic correlations, without

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

explicit mechanisms to ensure precise geometric and relational grounding required for complex road reasoning tasks [11, 12, 13].

At the same time, VLMs offer a unique opportunity. Unlike modular approaches, which rely on fixed schemas and often compress road structure into predefined, vectorized representations (HD maps, road graphs), VLMs support open-vocabulary reasoning and capture long-tail, semantically rich scene details such as faded markings, temporary changes, or ambiguous road configurations [14, 15, 16]. We hence identify a fundamental representation gap: traditional modular systems are structurally precise but semantically rigid, whereas VLMs today are semantically flexible but structurally ungrounded. The challenge is therefore not only to recover the geometrical and topological "substrate" of road understanding in modular approaches, but to extend it beyond the constraints of traditional representations by combining geometric precision with semantic flexibility.

To address this, we introduce the **Combined Road Substrate (CRS)**, a graph-grounded supervision framework for visual reasoning in road scenes. By enforcing a fixed compositional structure while allowing open-vocabulary semantic instantiation—and crucially, by making both directly queryable in a single graph representation—the CRS ensures that semantic expressivity remains grounded in, and constrained by, the underlying road structure. Concretely, reasoning tasks are formulated as queries over the spatio-temporal graph, allowing extraction of precise, compositional, and verifiable supervision signals in natural language in the form of question-answer pairs with hard negative mining. In moving beyond simple QA extraction, the CRS framework enables two key technical capabilities. First, it introduces a "grounding for free" mechanism via *Recursive Uniqueness*, which ensures that every generated reasoning task remains logically traceable to specific map elements with unambiguous spatial anchoring. Second, it allows for procedural extraction of the Chain-of-Thought (CoT), providing a verifiable supervision trace for explicit, graph-grounded reasoning.

We show that while even massive closed-source models [17, 18, 19] struggle to reason about complex road scenes, training on a remarkably small set (20–80) of CRS-enriched scenes enables VLMs to overcome inherent struggles with road understanding, achieving stable gains in compositional reasoning tasks of varying reasoning depth. Leveraging the verifiable CoT traces provided by CRS, we further analyze model behavior and localize reasoning failures to specific stages of the reasoning process. This reveals a systematic shift in failure modes: whereas standard models fail predominantly at relational reasoning (*i.e.*, navigating the scene structure), failures in models trained with CRS are no longer caused primarily by lack of scene understanding, but reduced to failures in attribute recognition (*i.e.*, visual perception). Taken together, these findings suggest that on road understanding questions provided by CRS, the absence of structured supervision limits current VLMs more than model scale. In summary, our main contributions are as follows:

- We introduce the **Combined Road Substrate (CRS)**, a graph-based representation that unifies geometric structure and open-vocabulary semantics in a jointly executable form, enabling grounded and compositional reasoning over road scenes.
- To make the CRS executable, we formulate reasoning tasks as **recursive queries over the graph**, enabling automatic generation of compositionally complex question–answer pairs with hard negatives. Grounding comes for free via *recursive uniqueness*, and verifiable CoT traces are derived directly from the graph in natural language.
- We demonstrate that state-of-the-art VLMs struggle with structured road understanding, and that training a smaller model on as few as 20–80 CRS-enriched scenes yields **substantial gains in compositional reasoning**, suggesting that CRS-style structured supervision can matter more than scale for compositional road reasoning tasks.

2 Related Work

Road understanding in modular AV stacks. Autonomous driving scene understanding has long relied on high-definition (HD) maps providing centimeter-accurate lane geometry, road topology, and semantic annotations [20, 21, 22]. Early approaches rasterized these representations into bird's-eye-view (BEV) feature maps [23, 24, 6], while subsequent work shifted towards predicting HD maps online as vectorized outputs [1, 2, 25]. Today, Transformer-based permutation-equivalent modeling is the dominant real-time methodology [3, 26]. While online rasterized or vectorized HD maps represent the geometry of road structure, they usually do not represent crucial relationships between road

elements needed for path planning, such as signs and traffic lights to lanes [4]. Different from HD map prediction, lane-topology prediction jointly predicts road geometry and the governance relationships between traffic controls and lanes [5, 27, 28, 29], enabling richer road reasoning. Complementing map geometry, scene graph representations encode both static road entities and dynamic actors (*i.e.*, vehicles, pedestrians) as nodes with relational edges, enabling joint reasoning over infrastructure and behavior [30, 31]. However, these structured representations over road scenes still need to be consumed by downstream modules for actionable road understanding.

Vision-language reasoning in autonomous driving. Language has recently emerged as a flexible interface for querying and reasoning about visual scenes, enabling models to express spatial relationships, object attributes, and compositional structure in natural language [32, 33, 34, 35, 36]. Seminal work has transferred this paradigm to autonomous driving, probing spatial relationships, object states, and navigational intent through QA-based benchmarks [37, 38]. Beyond single-step perception queries, subsequent work couples vision-language models with structured reasoning over driving scenes - modeling perception, prediction, and planning - as connected question-answer nodes [39], or integrates large VLMs directly with motion planners [40, 41, 42, 43]. Most recently, work has shifted toward explicit chain-of-thought and causal reasoning benchmarks [44, 36], including retrieval-augmented approaches that ground reasoning chains in visual crops [45]. While these approaches successfully build a bridge between language and driving, their primary focus often lies in action-oriented tasks rather than explicitly grounding the rich language-guided reasoning in the underlying geometric and topological substrate.

Representations for bridging structure and language. Beyond driving-specific approaches, a broader line of work has explored how to combine structured representations with language as a means to derive expressive supervision for grounded reasoning. A central challenge is balancing semantic flexibility with structural grounding: learned captioning [46] and vision-language models trained on image-text pairs [14, 47, 9, 38] are expressive, but remain weakly grounded and hard to scale, while supervision derived from structured representations - such as template-based QA generation [34, 37] - ensures verifiability, but constrains linguistic naturalness and diversity. For reasoning tasks specifically, recent work leverages procedural generation to produce datasets with controlled reasoning complexity and strong guarantees such as leakage resistance [48, 49], demonstrating the benefits of structured, executable supervision. Extending this idea to visual domains, scene graphs provide a structured yet expressive representation of entities and their relationships, and have been used to ground visual understanding in VLMs [50, 51, 52, 53, 54, 55]. Together, these lines of work suggest that structured, controllable representations are key for probing compositional reasoning, but lack a flexible language interface that tightly couples semantics with the underlying structure.

3 The Combined Road Substrate

To couple the compositional structure required for reasoning (*e.g.*, "the lane that is left of the ego lane") with the semantic flexibility needed to capture real-world complexity in road reasoning, we introduce the **Combined Road Substrate**, a graph-based scene representation that makes structure and semantics jointly executable. Concretely, our framework consists of four key components: (i) **graph primitives** that define the minimal *formal structure* of the graph representation, (ii) **canonical operators** that act as an interface between the graph primitives and the open vocabulary language space that provides the content of the graph, (iii) **well-posedness constraints** that ensure linguistic references to graph elements are unambiguous, hence reasoning over the graph is executable; and (iv) a **query instantiation mechanism** that defines reasoning tasks as structured answer retrieval through search queries over the graph, while mining hard negatives and chain-of-thought reasoning traces within the same framework.

3.1 Graph Primitives

We represent each scene as a directed graph over time steps $t \in [0, ..., T]$, $G = \{G_0, \dots, G_T\}$ with $G_t = (N_t, E_t)$, where N_t is the set of nodes and E_t the set of directed edges at time t. Each node $n \in N_t$ has a type $\tau(n)$ and a set of properties $P_t(n) = \{p_t^1, p_t^2, \dots\}$, where a property is a tuple $p_t^i = (k^i, y^i(t))$, with $k^i \in \mathcal{K}$ its property key and $y^i(t) \in \mathcal{Y}$ its corresponding value, which may be static for invariant properties ($y^i(t) = y_0$). Geometric grounding is represented as a special property $p_t^{\text{geo}} \in P_t(n)$ with values in the image-space $\mathbb{R}^2$ or real world coor-

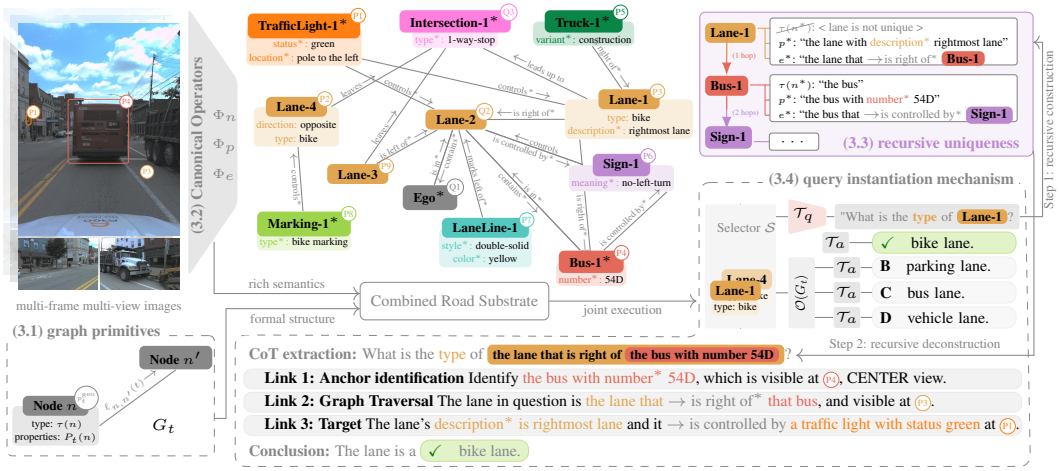


Figure 1: Language descriptions of road scenes are matched to graph primitives through canonical operators to build a semantically rich graph. This Combined Road Substrate is executable and produces questions, answers, decoys and CoT from the same underlying scheme.

dinates $\mathbb{R}^3$. Finally, each edge $e_t \in E_t$ is defined as $e_t = (n_t, n'_t, \ell_{n,n'}(t))$, where $n_t, n'_t \in N_t$ are the source and target nodes, and $\ell_{n,n'}(t)$ is a (possibly time-dependent) edge label. This formulation provides a minimal structural backbone upon which language-grounded semantics are later imposed.

3.2 Canonical operators

A central design goal of CRS is to avoid defining a priori what real-world objects (lanes, vehicles, crossings, etc.) are mapped to the graph. Instead, types, property keys and values, and edge labels should be freely expressible within the open-language space. However, without a well-specified language-structure interface that defines how open-vocabulary statements translate into their graph counterparts, the representation risks becoming inconsistent, ambiguous, and ultimately non-executable. To resolve this, we impose structure not on *what* can be expressed, but on *how* it must be expressed through three *canonical operators* Φ_n, Φ_p, Φ_e: An open language graph element is valid only if it produces a grammatical and semantically well-typed English sentence under one of the three canonical operators. For node types $\tau(n)$, we define

$$\Phi_{\text{n}}(\tau(n)) = \text{``There exists a <}\tau(n)\text{>''}. \tag{1}$$

Thus, a node type is valid if it completes an existential statement denoting an identifiable entity or group of objects, *e.g.*, ‘‘`There exists a traffic jam`’’. For properties, each $p_t^i = (k^i, y^i(t)) \in P_t(n)$ must satisfy

$$\Phi_{\text{p}}(n, k^i, y^i(t)) = \text{``The <}k^i\text{> of <}\tau(n)\text{> is `<}y^i(t)\text{>' ''}. \tag{2}$$

This rules out ill-typed encodings such as `is_closed_for_construction = true`, since ‘‘`The is_closed_for_construction of the lane is 'true' ''` is not a well-formed attribute statement. Instead, the same information should be represented as, for example, `status = closed_for_construction`. For relations, each edge $e = (n, n', \ell_{n,n'}(t))$ must satisfy

$$\Phi_{\text{e}}(n, n', \ell_{n,n'}(t)) = \text{``The <}\tau(n)\text{> <}\ell_{n,n'}(t)\text{> the <}\tau(n')\text{>''}. \tag{3}$$

Thus, relation labels must form meaningful directed relational statements, which excludes forms such as `'left of'` or `'controlling'`, but allows `'is left of'` or `'controls'`. By enforcing a fixed structural form while leaving the semantic space unconstrained, the canonical schema decouples compositional structure from linguistic expressivity. As a result, the graph remains consistent and queryable without restricting the richness or extensibility of its semantic content.

3.3 Well-definedness constraints

To make reasoning over G_t executable, linguistic references to nodes must be unambiguous and the relevant parts of the graph sufficiently specified. This is non-trivial under our open-vocabulary

design: multiple nodes may share the same type or properties (e.g., ‘‘`the traffic light with status ‘red’ ’’` may refer to multiple nodes), making language-based references ambiguous. Furthermore, since we do not constrain the space of potential node types, the graph operates under an *open-world assumption*, where absence does not imply non-existence but rather unknown state, and queries requiring exhaustive knowledge (*e.g.*, ‘‘`How many traffic lights are there?`’’) could potentially be ill-posed. We address this through two conditions: *uniqueness* and *completeness*.

(a) Uniqueness. A node in the graph is well-defined only if it admits a *unique* descriptor for the corresponding object in the scene. A descriptor d is a language expression derived from G_t that refers to the real-world object n represented via node type-, property-, or edge-based descriptions (*e.g.*, ‘‘`the traffic light`’’, ‘‘`the traffic light with status ‘red’ `’’, ‘‘`the traffic light that controls the ego lane`’’). To ensure identifiability, we introduce *uniqueness anchors* (denoted by *), which signal that the marked node, property or edge can be used to produce a descriptor that refers to *exactly one* object in the image. We define uniqueness at

- **node-level,** where the node's type $\tau(n^\star)$ alone is sufficient to identify the object in the scene (*e.g.*, the ‘only’ sign),
- **property-level**, where one of its properties $p^\star$ uniquely identifies the node (*e.g.*, the ‘only’ sign with meaning “no left turn”), and
- **edge-level,** $e^\star$, where a node is uniquely identified through its relation to another node (*e.g.*, the ‘only’ sign that controls the lane.)

Descriptors are constructed recursively by iterating over unique anchors, until uniqueness is achieved; if necessary, geometric grounding p_t^{geo} is used for disambiguation. Objects that cannot be uniquely identified are excluded from well-defined queries. Further details are provided in Section A.3.

(b) Completeness. To assure that queries about completeness or cardinality of nodes remain compatible with our open-world assumption, we define a completeness indicator at *node-type level*. For a type $\tau(n)$, let $c_t(\tau) \in \{0, 1\}$ denote whether all instances of type τ that are visible in the input image at time t are also present in N_t. A query q that requires exhaustive enumeration over nodes of type τ is then considered valid only if $c_t(\tau) = 1$. Taken together, the first three components of the CRS framework yield a representation that is structurally well-defined yet remains semantically expressive, while ensuring that objects in the scene can be uniquely and unambiguously associated with individual nodes in the graph.

3.4 The query instantiation mechanism

At the core of CRS is the query instantiation mechanism, which turns the graph representation into an executable framework, hence formulates reasoning tasks as structured computations over the graph. Each query extracts a question, its correct answer, and a set of plausible decoys, all expressed in natural language. Additionally, CoT traces are automatically derived in the same framework, while reasoning depth is controlled as a hyperparameter of the method.

Query structure. Each query is defined by four components: (i) a selection operator $\mathcal{S}$ that retrieves a set of target nodes and associated attributes or relations from G_t, (ii) a question template $\mathcal{T}_q$ that maps the selected elements to a natural language question, (iii) an answer template $\mathcal{T}_a$ that renders the correct answer, and (iv) a perturbation operator $\mathcal{O}(G)$ that produces controlled variations of the graph or its attributes. To generate hard negatives, applying $\mathcal{T}_a$ to $\mathcal{O}(G)$ yields semantically plausible but incorrect answer candidates.

While a virtually infinite set of queries can be defined to inquire different aspects of the graph, we identify five fundamental categories: queries targeting node properties, inter-node relations, counting and comparison, of nodes, and existence/positional queries. For our study, we instantiate 19 different queries targeting a broad range of graph properties (lane type, line color, counting of lanes and crossings, relationships between lanes, etc.). While all query types used in this study are listed in Section A.1 of the supplementary material, we discuss an example of property-related queries below — together with edge-related queries the most fundamental operation on the graph. Furthermore, they are relatively simple and hence an important means for model analysis (see Section 4).

Let us assume we have a road scene as illustrated in Figure 1 with the CRS. Since property-related queries inquire the value of node properties for nodes of a certain type, our selector can be written as

$$\mathcal{S}(N_t, \hat{\tau}, \hat{k}) = \{n \in N_t, (k^i, y^i(t)) \in P_t(n) \mid \tau(n) = \hat{\tau} \cap k^i = \hat{k}\}, \tag{4}$$

while the question template is $\mathcal{T}_q$ ="What is the <$\hat{k}$> of the $d^*(n)$"? The answer template simply becomes the property value, $\mathcal{T}_a$ ="$y(t)$". If we select $\tau(n) =$ `lane` and $\hat{k} =$ `type`, $\mathcal{S}$ will extract exactly two matching nodes from the graph (`Lane-1`, `Lane-4`), while $\mathcal{T}_a$ yields "bike lane" for both. For decoys, $\mathcal{O}$ perturbs the graph by substituting other potential types ("bus lane", etc.). The semantic diversity (and variation) of our framework is then activated through the uniqueness mechanism inside $\mathcal{T}_q$. When resolving for $d^*(n)$ (`Lane-1` in Figure 1), we get a variety of unique options, ranging from ‘‘`What is the type of the lane that contains the bus with number 54D?`’’ to ‘‘`What is the type of the right curb lane?`’’, for the same query and node.

Reasoning depth and chain-of-thought. Because queries refer to objects in the scene through the unique descriptor present in $\mathcal{T}_q$, $\mathcal{T}_a$, or both, the compositional complexity of a query is directly controlled by the maximum descriptor depth, *i.e.*, a hyperparameter that defines how many neighboring nodes can be traversed to derive unique descriptors. Concretely, zero reasoning hops are necessary to resolve ‘‘`the right curb lane`’’, while ‘‘`the lane that contains the bus with number 54D`’’ requires one. The consequences of this explicit dependency are threefold: Not only are semantically diverse multi-hop questions created automatically and with controlled reasoning depth, but a chain-of-thought reasoning trace can also be generated directly from the graph by traversing the descriptor in reverse order and grounding the nodes present in the descriptor chain. While - as for the other parts of the query - individual queries can combine these links in different ways, we identify three primary links in the reasoning chain:

1. **Anchor identification:** We resolve the outermost element of $d^*(n)$ to a unique node $n_a \in N_t$, guaranteed by the uniqueness constraint. This provides an initial grounding in image space via $p_t^{\text{geo}}(n_a)$. For the example in Figure 1, this would consist of identifying and grounding ‘‘`the bus with number 54D`’’.
2. **Graph traversal:** We iteratively resolve the nested structure of $d^*(n)$ by traversing edges in G_t, producing a sequence of nodes $(n_a, \ldots, n)$ that terminates at the target node (*i.e.*, ‘‘`the lane that contains the bus with number 54D`’’).
3. **Target extraction:** To help ground the target node, we first extract a brief description by combining - unique and non-unique - properties and edges of n to also visually ground the target. Finally, we retrieve the attribute or relation from n needed for the answer.

Each intermediate node in the traversal is grounded via its geometric property, ensuring that the entire reasoning trace is spatially anchored. This formulation yields three key properties. First, the CoT trace is *deterministic*, as it is uniquely defined by the query and the graph. Second, it is *fully grounded*, since each step corresponds to explicit nodes and relations in G_t. Third, it is *compositional* and can be potentially repeated multiple times for queries such as counting or comparison, where the procedure is applied once per enumerated element. Importantly, this CoT arises for free alongside question and answer generation, requiring no additional annotation effort. Implementation details about the three links and the specific CoT instantiations per query are provided in Section A.2.

3.5 Data sources and graph construction

To test the effectiveness of our framework, we instantiate CRS by integrating standard geometric and semantic representations from modular driving stacks, including HD maps (lane geometry and layout), topological annotations (lane connectivity and control relations), and object annotations (dynamic agents and scene elements), and subsequently enrich them with natural language. Concretely, we use the front-facing camera views from Argoverse2 [21] as visual input, and combine the provided HD maps with the topological structure from OpenLane-V2 [4] to populate our graph. These sources are mapped into CRS using the canonical operators: map elements become nodes (Φ_n), attributes are encoded as properties (Φ_p), and relationships (*e.g.*, adjacency, control) constitute edges (Φ_e).

Although these representations are available and compatible with the CRS, naive aggregation of all information would result in a graph that is overly dense with respect to number of nodes, poorly aligned with the selectiveness of human visual reasoning, and lacking in semantic richness. Rather

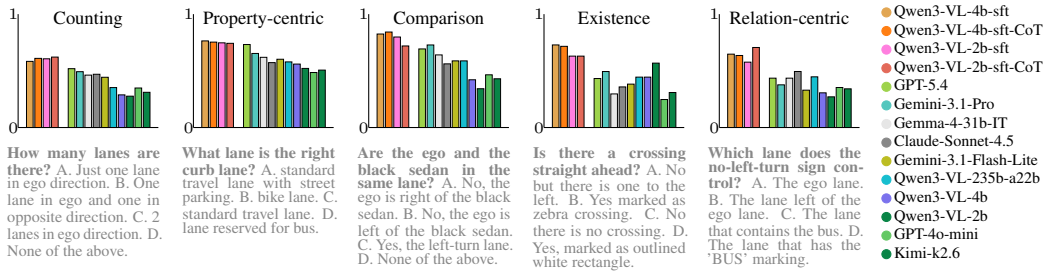


Figure 2: Model performance by bucket with example questions.

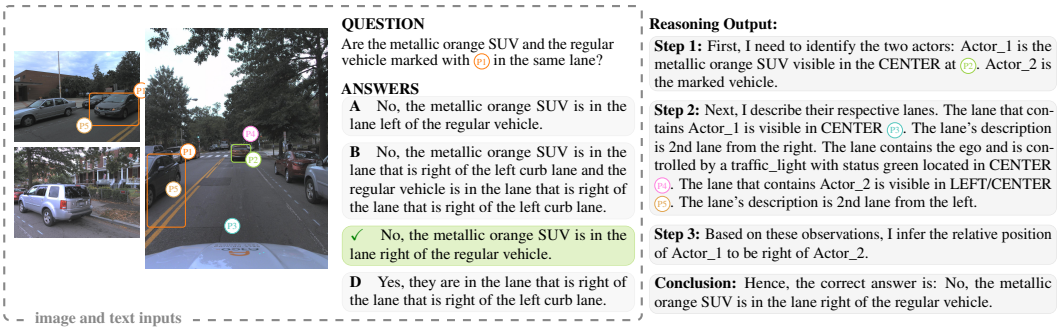


Figure 3: Example output of the Qwen3-VL-4b model with CoT (additional examples in Section A.9)

than directly aggregating all data, we construct graphs through a *selective enrichment* process with human oversight. Starting from the initial graph, expert human annotators (i) *filter* for salience by removing irrelevant or non-visible elements, (ii) *add* entities that are important for scene understanding, including objects beyond existing annotations, (iii) *enrich* nodes with additional properties in natural language, and (iv) *introduce* new relations that capture context-specific interactions. This process also establishes uniqueness and completeness required for executable reasoning.

Using this pipeline, we construct 80 CRS graphs, from which we generate 23k training samples via 19 queries on a temporal window of 4 frames at 2 Hz. We also create a held-out validation set of 1000 expert-verified samples from 120 distinct scenes following [56], subsampling at most one sample per query and scene and computing confidence intervals with scene-level cluster bootstrap, to account for statistical dependence between questions from the same scene. Further details on graph construction, as well as human annotation and quality assurance effort are provided in Section A.4 and Section A.5. The graph creation tool will be released together with the code and data.

4 Experiments

We evaluate the effectiveness of CRS-based supervision by fine-tuning vision-language models on graph-derived training data and analyzing their behavior across multiple axes. In particular, we focus on (i) overall performance gains, (ii) data efficiency and scaling behavior, and (iii) robustness under increasing reasoning complexity. We fine-tune models from the Qwen family using full-parameter training on 4×H100 GPUs for 5 epochs, with a learning rate of 4×10^{-5} and cosine decay. The exact training and evaluation prompt is reported in Section A.7.

Performance comparison with and without CRS supervision. We first evaluate the impact of CRS supervision by comparing models with and without graph-grounded supervision. To contextualize these gains, we additionally evaluate a range of state-of-the-art open- and closed-weight VLMs, which serve as a reference point for current capabilities on structured road reasoning. To disentangle perception from reasoning, we partition the evaluation data into *perception-like* and *reasoning-heavy* samples based on the hardness of decoy answers. In perception-like tasks, the primary challenge lies in identifying the correct object or attribute from the image, after which answer selection is largely straightforward (*e.g.*, "What is the status of the traffic light that controls the crossing to the right of the intersection?"—"Walk / Stop / Countdown"). In contrast, reasoning-heavy tasks require

Table 1: Accuracy is reported as decimals with 95% scene-clustered CI. Accuracy for additional models is provided in Section A.8. *We randomly select 300 samples, which are given to 4 human driving experts.

Model	Total	Perception-like	Reasoning-heavy
Human expert*	0.955 (0.943,0.967)	-	-
GPT-4o-mini [57]	0.417 (0.383, 0.450)	0.445 (0.408, 0.480)	0.305 (0.237, 0.374)
Gemini-3.1-Flash-Lite [17]	0.519 (0.485, 0.550)	0.554 (0.520, 0.590)	0.375 (0.314, 0.438)
Claude-Sonnet-4.5 [19]	0.521 (0.484, 0.555)	0.548 (0.513, 0.581)	0.410 (0.342, 0.473)
Gemini-3.1-Pro [17]	0.584 (0.549, 0.618)	0.609 (0.571, 0.649)	0.480 (0.415, 0.541)
GPT-5.4 [18]	0.621 (0.593, 0.654)	0.667 (0.636, 0.704)	0.435 (0.374, 0.495)
Molmo2-8B [11]	0.405 (0.378, 0.435)	0.410 (0.378, 0.441)	0.385 (0.327, 0.443)
Kimi-k2.6 [58]	0.417 (0.385, 0.451)	0.439 (0.404, 0.473)	0.330 (0.262, 0.400)
Qwen3-VL-30b-a3b-Thinking [59]	0.460 (0.422, 0.496)	0.484 (0.445, 0.525)	0.360 (0.296, 0.428)
Qwen3-VL-235b-a22b-Instruct [59]	0.499 (0.467, 0.531)	0.512 (0.476, 0.546)	0.445 (0.380, 0.505)
Gemma-4-31b-IT [60]	0.539 (0.502, 0.577)	0.576 (0.536, 0.620)	0.390 (0.325, 0.455)
Qwen3-VL-2b-Instruct [59]	0.420 (0.392, 0.450)	0.429 (0.395, 0.461)	0.385 (0.317, 0.459)
+ *supervised fine-tune*	0.692 (0.658, 0.727)	0.715 (0.677, 0.753)	0.600 (0.533, 0.670)
+ *supervised fine-tune* + *CoT*	0.696 (0.664, 0.727)	0.715 (0.680, 0.750)	0.620 (0.556, 0.683)
Qwen3-VL-4b-Instruct [59]	0.443 (0.410, 0.476)	0.466 (0.429, 0.501)	0.350 (0.283, 0.427)
+ *supervised fine-tune* (− images)	0.587 (0.549, 0.625)	0.586 (0.545, 0.630)	0.590 (0.518, 0.656)
+ *supervised fine-tune*	*0.711* (0.672, 0.746)	*0.718* (0.679, 0.759)	**0.680** (0.605, 0.751)
+ *supervised fine-tune* + *CoT*	**0.713** (0.681, 0.746)	**0.722** (0.683, 0.759)	*0.675* (0.616, 0.734)

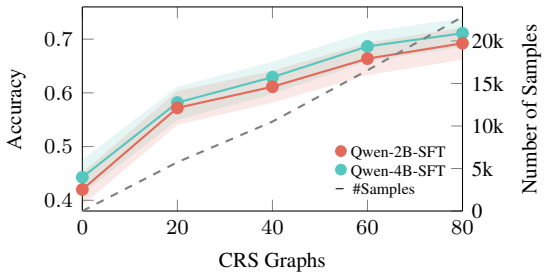


Figure 4: Accuracy over number of CRS-scenes, with sample counts on the secondary axis.

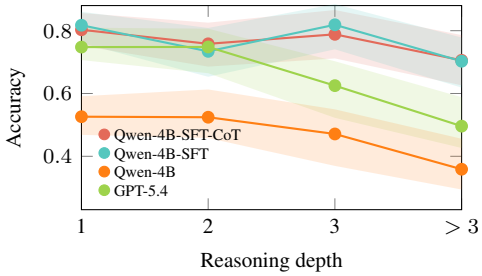


Figure 5: Accuracy by reasoning depth with 95% CI. Additional results in Section A.8.

compositional grounding: models must resolve nested descriptors, identify relevant entities through relational structure, and perform multi-step reasoning (*e.g.*, "Which lane is the ego in?"—"The lane that is left of the emergency vehicle's lane. / The lane that contains the arrow. / The right curb lane. / None of the above"). A breakdown of the query templates in each category and details on the question distribution are provided in Section A.1 and Section A.6.

As shown in Table 1, performance across all models degrades substantially on reasoning-heavy samples, indicating that structured reasoning remains a key challenge. After fine-tuning with CRS-derived data, we observe substantial improvements in overall accuracy and robustness to reasoning complexity. Notably, even small models exhibit strong gains, particularly on reasoning-heavy tasks (see Figure 2 and Figure 3). We further train a baseline model without visual input (cf. − *images* in Table 1) on our CRS-derived data, and find that while the model captures some road structure—because roads inherently contain structure—there remains a significant performance gap compared to vision-augmented models and human performance, highlighting the necessity of visual grounding. A detailed investigation of which aspects of the road the no-image baseline is able to learn, as well as an ablation of the role of the decoys are provided in Section A.8.

Data efficiency and scaling. Next, we analyze how performance scales with the number of CRS graphs used for training by varying the number of scenes and measuring downstream accuracy. As shown in Figure 4, performance improves consistently with additional data, while exhibiting diminishing returns at larger scales. Notably, training on as few as 40 scenes already surpasses the performance of strong closed-source baselines. This highlights the data efficiency of CRS supervision:

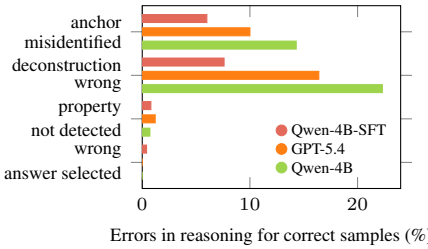


Figure 6: Distribution of all wrong links in the reasoning chain for samples where the model's final answer is correct.

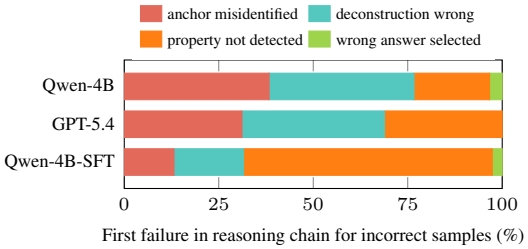


Figure 7: Distribution of first wrong link in the reasoning chain for samples where the model's final answer is incorrect.

even in the low-data regime, the compositional structure of graph-derived queries provides dense and informative training signals, while creating a large training set from just a few scenes.

Reasoning depth analysis. A central question is whether CRS supervision improves robustness to increasing reasoning complexity. Leveraging the programmatic structure of CRS, we quantify reasoning depth directly from the graph by measuring the number of compositional steps required to resolve a query. While the maximum descriptor depth for individual nodes is limited to two, queries may involve multiple uniquely referenced objects across questions, answers, and distractors, resulting in a broad spectrum of effective reasoning depths. To evaluate the impact of CRS supervision, we partition the evaluation set by reasoning depth, *i.e.*, number of hops on the graph required to solve the task. As shown in Figure 5, models trained with CRS supervision maintain stable performance as reasoning depth increases, in contrast to both the base model and strong closed-source baselines, which exhibit a pronounced degradation (see Section A.8 for additional results). This indicates that CRS supervision enables models to reliably decompose nested descriptors and execute multi-step relational reasoning over the scene.

Chain-of-thought analysis. Building on this, the CRS framework provides a unique opportunity to trace performance differences back to specific reasoning failures. While CoT supervision does not necessarily improve aggregate accuracy (cf. Table 1), it enables fine-grained analysis of where and how models fail. For this analysis, we sample 250 questions from the validation set following the property-query schema (see Section 3.4). These questions target relatively simple properties—such as lane direction, marking type, or color—but require a well-defined sequence of reasoning steps. Importantly, each question can only be solved by completing the three links in the reasoning chain (cf. Section 3.4): (i) Localizing the anchor object mentioned in the question, *i.e.*, the last step in a multi-hop descriptor, (ii) resolving the relational description to identify the target node and (iii) retrieving the relevant property. Finally, the last step consists of (iv) choosing the correct option.

We inspect, for both correct and incorrect predictions, which of these steps are executed correctly. For correct answers, this provides a measure of reasoning consistency: cases where the answer is correct despite incorrect intermediate reasoning indicate spurious correlations. For incorrect answers, the first failed step reveals the origin of the error. For correctly answered questions, our model exhibits higher consistency between reasoning and final predictions (Figure 6). More importantly, the analysis of incorrect predictions reveals a qualitative difference: in our model, the first failure most often occurs at the property extraction stage, after successfully resolving the object through multi-hop reasoning (Figure 7). In contrast, the closed-source model predominantly fails at earlier stages of anchor identification and descriptor decomposition, indicating difficulties in navigating the scene structure in the first place. This suggests that models fine-tuned with CRS-data have internalized the compositional structure of the road scene.

Generalization. Finally, we consider 3 axes of generalization for our models: generalization (i) across scenes, (ii) across linguistic patterns, and (iii) across different question types. (i) In constructing the validation split used for the experiments, we already test for generalization across different scenes, since our expert-verified samples stem from scenes fully disjoint both geographically and temporally from the training scenes following [56], with at most one sample per query type per scene to avoid inter-question redundancy. With respect to language and formulation overfitting (ii), we paraphrase 300 validation questions with an LLM (Gemini-2.5-Flash [61], same graph,

Table 2: Model performance on original formulation and after paraphrasing 300 randomly chosen samples with Gemini-2.5-Flash [61]. Accuracy reported with 95% CI with McNemar's test.

Model	Original Accuracy	Accuracy after paraphrasing	Δ	p (McNemar)
QwenVL-4B-SFT	0.717 (0.663, 0.767)	0.680 (0.627, 0.733)	-3.67 pp	0.13
QwenVL-4B-SFT+CoT	0.720 (0.670, 0.770)	0.683 (0.630, 0.737)	-3.67 pp	0.12

Table 3: Zero-shot performance when training on all but that held-out bucket, compared to the Qwen3-VL-4b-Instruct [59] baseline. Accuracy reported with 95% CI with McNemar's test.

Held-out bucket	n	Held-out Accuracy	Baseline Accuracy	Δ	p (McNemar)
Comparison	114	0.649 (0.553, 0.737)	0.421 (0.333, 0.509)	$+22.81$ pp	0.0001
Counting	264	0.383 (0.326, 0.439)	0.288 (0.231, 0.345)	$+9.47$ pp	0.0187
Relational	85	0.400 (0.294, 0.506)	0.306 (0.212, 0.400)	$+9.41$ pp	0.2295
Existence	81	0.482 (0.370, 0.593)	0.444 (0.333, 0.556)	$+3.70$ pp	0.7011
Properties	470	0.568 (0.526, 0.613)	0.560 (0.517, 0.602)	$+0.85$ pp	0.8312
Rel.+Comp.+Exist.	280	0.532 (0.475, 0.593)	0.393 (0.332, 0.454)	$+13.93$ pp	0.0005

same underlying generated questions and answers, reworded surface form) and re-evaluate our Qwen-4B-SFT and Qwen-4B-SFT-CoT models. The results are provided in Table 2 with 95% CI and McNemar p values, with the modest decrease not being statistically significant, suggesting performance is not tied to CRS's specific template wording. Lastly, to probe (iii) generalization to unseen question types, we retrain the Qwen-4B-SFT model on four of the five query buckets in turn (leaving only isolated question types out would result in very small per-type evaluation samples with low statistical significance): we exclude each bucket entirely from the 23k training samples and evaluate zero-shot on the held-out bucket against the untrained baseline, using McNemar's test on the paired per-item outcomes (Table 3). We observe promising zero-shot generalization across `Comparison` and `Counting` buckets, while `Relational` and `Existence` alone are directionally positive (+9.4 pp, +3.7 pp) but not individually significant given the small sample sizes of n=81–85 (p=0.23, p=0.70). The last row is a separate retraining with the three smallest buckets removed simultaneously, evaluated on their union, not an aggregation of the rows above. This is the strictest of our settings: the remaining training set contains no reasoning-heavy query type at all, yet still improves significantly over baseline. The only bucket that shows no generalization is `Properties`, which we attribute to smaller headroom from a substantially higher baseline. Note that every retrained model sees strictly less training data than the full model, so these gains are conservative.

5 Discussion and conclusion

This work introduces the Combined Road Substrate (CRS), a graph-grounded framework that unifies geometric structure and open-vocabulary semantics in a jointly executable representation. By formulating reasoning as queries over this substrate, CRS enables the automatic generation of grounded, compositional supervision signals, yielding substantial gains in structured road reasoning even in low-data regimes. Our analysis further demonstrates that structured supervision fundamentally shifts model behavior, redirecting failures from relational reasoning to localized perceptual errors.

Limitations and future work. While CRS improves structural reasoning, performance remains bounded by the quality of underlying visual perception, indicating that further gains depend on advances in perception models. CRS construction relies on selective human curation by design, which adds annotation overhead, and assumes HD-map-like topology; automating abstraction and enrichment is future work. Our data stem from a single source dataset with limited geographic diversity, and training and evaluation share one query grammar; while held-out question types and paraphrases show no collapse, models may still overfit to the CRS schema. Reasoning traces may look interpretable without being faithful and must not be taken as safety evidence. Lastly, while this work focuses on the substrate of road understanding, we do not explicitly tackle downstream driving performance. However, we believe that CRS provides a promising grounded intermediate representation for future vision-language-action (VLA) driving systems, while executable and compositional structure may offer a natural interface between semantic reasoning and policy learning — an important direction left for future work.

Acknowledgements and Disclosures. The research work was partially funded by the Swedish Foundation for Strategic Research (SSF) under the project DeltaMap (ID22-0045) and by a gift from Volkswagen Group of America. This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Agreement Number 00011869. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA). We thank Patric Jensfelt, Rafael Valencia and Vahdat Abdelzad for their feedback on this work.

References

[1] Qi Li, Yue Wang, Yilun Wang, and Hang Zhao. Hdmapnet: An online hd map construction and evaluation framework. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 4628–4634. IEEE, 2022.

[2] Yicheng Liu, Tianyuan Yuan, Yue Wang, Yilun Wang, and Hang Zhao. Vectormapnet: End-to-end vectorized hd map learning. In *International conference on machine learning*, pages 22352–22369. PMLR, 2023.

[3] Bencheng Liao, Shaoyu Chen, Xinggang Wang, Tianheng Cheng, Qian Zhang, Wenyu Liu, and Chang Huang. Maptr: Structured modeling and learning for online vectorized hd map construction. In *International Conference on Learning Representations*, 2023.

[4] Huijie Wang, Tianyu Li, Yang Li, Li Chen, Chonghao Sima, Zhenbo Liu, Bangjun Wang, Peijin Jia, Yuting Wang, Shengyin Jiang, et al. Openlane-v2: A topology reasoning benchmark for unified 3d hd mapping. *Advances in Neural Information Processing Systems*, 36:18873–18884, 2023.

[5] Tianyu Li, Li Chen, Huijie Wang, Yang Li, Jiazhi Yang, Xiangwei Geng, Shengyin Jiang, Yuting Wang, Hang Xu, Chunjing Xu, et al. Graph-based topology reasoning for driving scenes. *arXiv preprint arXiv:2304.05277*, 2023.

[6] Bin Yang, Ming Liang, and Raquel Urtasun. Hdnet: Exploiting hd maps for 3D object detection. In *CoRL*, 2018.

[7] Sergio Casas, Abbas Sadat, and Raquel Urtasun. Mp3: A unified model to map, perceive, predict and plan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14403–14412, 2021.

[8] Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, et al. Planning-oriented autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 17853–17862, 2023.

[9] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

[10] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

[11] Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, et al. Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 91–104, 2025.

[12] Liang Chen, Weichu Xie, Yiyan Liang, Hongfeng He, Hans Zhao, Zhibo Yang, Zhiqi Huang, Haoning Wu, Haoyu Lu, Yiping Bao, et al. Babyvision: Visual reasoning beyond language. *arXiv preprint arXiv:2601.06521*, 2026.

[13] Stanford HAI AI Index Team. Technical performance. Technical report, Stanford University Institute for Human-Centered Artificial Intelligence, 2026. Accessed: April 30, 2026.

[14] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[15] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022.

[16] Sravanti Addepalli, Ashish Ramayee Asokan, Lakshay Sharma, and R Venkatesh Babu. Leveraging vision-language models for improving domain generalization in image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23922–23932, 2024.

[17] The Gemini Team. Gemini 3.1 Pro: A smarter model for your most complex tasks. `https://blog.google/innovation-and-ai/models-and-research/gemini-models/gemini-3-1-pro/`, February 2026.

[18] OpenAI. Introducing GPT-5.4. `https://openai.com/index/introducing-gpt-5-4/`, March 2026. OpenAI Blog.

[19] Anthropic. Claude sonnet 4.5. `https://www.anthropic.com`, September 2025. Large language model, released September 29, 2025. Model string: claude-sonnet-4-5-20250929.

[20] 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*, pages 11621–11631, 2020.

[21] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan, Peter Carr, and James Hays. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks 2021)*, 2021.

[22] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[23] Brady Zhou and Philipp Krähenbühl. Cross-view transformers for real-time map-view semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13760–13769, 2022.

[24] Heiko G. Seif and Xiaolong Hu. Autonomous driving in the icity—hd maps as a key challenge of the automotive industry. *Engineering*, 2:159–162, 2016.

[25] Juyeb Shin, Hyeonjun Jeong, Francois Rameau, and Dongsuk Kum. Instagram: Instance-level graph modeling for vectorized hd map learning. *IEEE Transactions on Intelligent Transportation Systems*, 26(2):1889–1899, 2025.

[26] Bencheng Liao, Shaoyu Chen, Yunchi Zhang, Bo Jiang, Qian Zhang, Wenyu Liu, Chang Huang, and Xinggang Wang. Maptrv2: An end-to-end framework for online vectorized hd map construction. *International Journal of Computer Vision*, 133(3):1352–1374, 2025.

[27] Tianyu Li, Peijin Jia, Bangjun Wang, Li Chen, Kun Jiang, Junchi Yan, and Hongyang Li. Lanesegnet: Map learning with lane segment perception for autonomous driving. In *ICLR*, 2024.

[28] Dongming Wu, Jiahao Chang, Fan Jia, Yingfei Liu, Tiancai Wang, and Jianbing Shen. Topomlp: A simple yet strong pipeline for driving topology reasoning. *ICLR*, 2024.

[29] Katie Z Luo, Xinshuo Weng, Yan Wang, Shuang Wu, Jie Li, Kilian Q Weinberger, Yue Wang, and Marco Pavone. Augmenting lane perception and topology understanding with standard definition navigation maps. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4029–4035. IEEE, 2024.

[30] Yunpeng Zhang, Deheng Qian, Ding Li, Yifeng Pan, Yong Chen, Zhenbao Liang, Zhiyao Zhang, Yingzong Liu, Jianhui Mei, Maolei Fu, Yun Ye, Zhujin Liang, Yi Shan, and Dalong Du. Graphad: Interaction scene graph for end-to-end autonomous driving. In James Kwok, editor, *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, pages 2422–2430. International Joint Conferences on Artificial Intelligence Organization, 8 2025. Main Track.

[31] Fabian Schmidt, Markus Enzweiler, and Abhinav Valada. Graphpilot: Grounded scene graph conditioning for language-based autonomous driving. *arXiv preprint arXiv:2511.11266*, 2025.

[32] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433, 2015.

[33] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6904–6913, 2017.

[34] Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6700–6709, 2019.

[35] Boyuan Chen, Zhuo Xu, Sean Kirmani, Brian Ichter, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14455–14465, 2024.

[36] Weizhen Wang, Chenda Duan, Zhenghao Peng, Yuxin Liu, and Bolei Zhou. Embodied scene understanding for vision language models via metavqa. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22453–22464, 2025.

[37] Tianwen Qian, Jingjing Chen, Linhai Zhuo, Yang Jiao, and Yu-Gang Jiang. Nuscenes-qa: A multi-modal visual question answering benchmark for autonomous driving scenario. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 4542–4550, 2024.

[38] Ana-Maria Marcu, Long Chen, Jan Hünermann, Alice Karnsund, Benoit Hanotte, Prajwal Chidananda, Saurabh Nair, Vijay Badrinarayanan, Alex Kendall, Jamie Shotton, et al. Lingoqa: Visual question answering for autonomous driving. In *European Conference on Computer Vision*, pages 252–269. Springer, 2024.

[39] Chonghao Sima, Katrin Renz, Kashyap Chitta, Li Chen, Hanxue Zhang, Chengen Xie, Jens Beißwenger, Ping Luo, Andreas Geiger, and Hongyang Li. Drivelm: Driving with graph visual question answering. In *European conference on computer vision*, pages 256–274. Springer, 2024.

[40] Thierry Deruyttere, Simon Vandenhende, Dusan Grujicic, Luc Van Gool, and Marie Francine Moens. Talk2car: Taking control of your self-driving car. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 2088–2098, 2019.

[41] Srikanth Malla, Chiho Choi, Isht Dwivedi, Joon Hee Choi, and Jiachen Li. Drama: Joint risk localization and captioning in driving. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 1043–1052, 2023.

[42] Tushar Choudhary, Vikrant Dewangan, Shivam Chandhok, Shubham Priyadarshan, Anushka Jain, Arun K Singh, Siddharth Srivastava, Krishna Murthy Jatavallabhula, and K Madhava Krishna. Talk2bev: Language-enhanced bird's-eye view maps for autonomous driving. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16345–16352. IEEE, 2024.

[43] Xiaoyu Tian, Junru Gu, Bailin Li, Yicheng Liu, Yang Wang, Zhiyong Zhao, Kun Zhan, Peng Jia, Xianpeng Lang, and Hang Zhao. Drivevlm: The convergence of autonomous driving and large vision-language models. In *Conference on Robot Learning (CoRL)*, 2024.

[44] Ming Nie, Renyuan Peng, Chunwei Wang, Xinyue Cai, Jianhua Han, Hang Xu, and Li Zhang. Reason2drive: Towards interpretable and chain-based reasoning for autonomous driving. In *European Conference on Computer Vision*, pages 292–308. Springer, 2024.

[45] Charles Corbière, Simon Roburin, Syrielle Montariol, Antoine Bosselut, and Alexandre Alahi. DRIVINGVQA: A dataset for interleaved visual chain-of-thought in real-world driving scenarios. In Vera Demberg, Kentaro Inui, and Lluís Marquez, editors, *Findings of the Association for Computational Linguistics: EACL 2026*, pages 3309–3333, Rabat, Morocco, March 2026. Association for Computational Linguistics.

[46] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2015.

[47] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023.

[48] Yang Zhou, Hongyi Liu, Zhuoming Chen, Yuandong Tian, and Beidi Chen. Gsm-infinite: How do your llms behave over infinitely increasing context length and reasoning complexity? *arXiv preprint arXiv:2502.05252*, 2025.

[49] Albert Gong, Kamilė Stankevičiūtė, Chao Wan, Anmol Kabra, Raphael Thesmar, Johann Lee, Julius Klenke, Carla P Gomes, and Kilian Q Weinberger. Phantomwiki: On-demand datasets for reasoning and retrieval evaluation. In *Forty-second International Conference on Machine Learning*, 2025.

[50] Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David Shamma, Michael Bernstein, and Li Fei-Fei. Image retrieval using scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3668–3678, 2015.

[51] Xu Yang, Kaihua Tang, Hanwang Zhang, and Jianfei Cai. Auto-encoding scene graphs for image captioning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10685–10694, 2019.

[52] Jiaxin Shi, Hanwang Zhang, and Juanzi Li. Explainable and explicit visual reasoning over scene graphs. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8376–8384, 2019.

[53] Vinay Damodaran, Sharanya Chakravarthy, Akshay Kumar, Anjana Umapathy, Teruko Mitamura, Yuta Nakashima, Noa Garcia, and Chenhui Chu. Understanding the role of scene graphs in visual question answering. *arXiv preprint arXiv:2101.05479*, 2021.

[54] Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *Advances in Neural Information Processing Systems*, 37, 2024.

[55] Yihe Deng, Chenchen Ye, Zijie Huang, Mingyu Derek Ma, Yiwen Kou, and Wei Wang. Graphvis: Boosting llms with visual knowledge graph integration. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 67511–67534. Curran Associates, Inc., 2024.

[56] Adam Lilja, Junsheng Fu, Erik Stenborg, and Lars Hammarstrand. Localization is all you evaluate: Data leakage in online mapping datasets and how to fix it. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22150–22159, 2024.

[57] OpenAI. GPT-4o mini: Advancing Cost-Efficient Intelligence. `https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/`, July 2024. OpenAI Blog.

[58] Moonshot AI. Kimi K2.6. `https://www.kimi.com/blog/kimi-k2-6.html`, April 2026. Moonshot AI Blog.

[59] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[60] Google DeepMind. Gemma 4: Our Most Intelligent Open Models. `https://deepmind.google/models/gemma/gemma-4/`, April 2026. Google DeepMind.

[61] The Gemini Team. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

A Supplementary Material

A.1 Query instantiation

Let $G_t = (N_t, E_t)$ denote the CRS scene graph at the queried frame and let $G_{t-w:t}$ denote the temporal window of size $w = 4$ used by temporal queries. $\tau(n)$ is the node type, $p_n = (k^i, y^i(t))$ is property k^i of node n with value $y^i(t)$, and $(n, n', \ell_{n,n'}(t)) \in E_t$ denotes a directed relation. $d(n)$ is the recursive unique descriptor selected by the recursive descriptor construction. Table 4 below details the 19 queries used in this work.

Table 4: High-level formalization of all query types instantiated in the CRS.

Type	**Selection operator** $\mathcal{S}(G_t)$	**Question template** T_q	**Answer template** T_a	**Perturbation procedure** $\mathcal{O}(G_t)$
`lane direction`	Select each visible lane l with $\tau(l) =$ `lane`, excluding lanes whose descriptor would already reveal the direction or a near-intersection turn-side relation. Return $(l, p_l[\texttt{direction}])$.	"What is the direction of $d^*(l)$?"	Render $p_l[\texttt{direction}]$ as either "same direction as ego" or "opposite direction".	Attribute perturbation: replace the true binary direction with the other direction value. Distractors are the alternate direction labels.
`line marking`	Select lane-line/lane pairs (m, l) with $\tau(m) =$ `lane_line`, $\tau(l) =$ `lane`, and $(m, l,$ `is left marking of`$)$ or $(m, l,$ `is right marking of`$) \in E_t$.	"What is the left/right lane line marking of $d^*(l)$?"	Render the full marking style $p_m[\texttt{mark_type}]$, *e.g.*the verbose form of a dashed/solid and color combination.	Attribute perturbation: substitute another complete lane-marking style from the marking vocabulary; optionally include "None of the above".
`line type`	Use the same selected lane-line/lane pair (m, l) as `line_marking`.	"What is the left/right lane line type of $d^*(l)$?"	Render the type component of $p_m[\texttt{mark_type}]$, *e.g.*solid, dashed, double solid, double dashed, or no visible line.	Attribute perturbation: hold the lane and side fixed but replace the type component with another plausible line type; optionally include "None of the above".
`line color`	Use the same selected lane-line/lane pair (m, l) as `line_marking`.	"What is the left/right lane line color of $d^*(l)$?"	Render the color component of $p_m[\texttt{mark_type}]$, *e.g.*, white or yellow, not visible if it is implicit.	Attribute perturbation: hold the lane and side fixed but replace the color component with another color from the lane-line vocabulary.
`existence of crossings`	For a single nearby intersection, iterate over canonical crossing locations $s \in$ {left of , right of, straight ahead on, at far side of} the intersection. Select all described crossing nodes c and check whether any visible c satisfies $p_c[\texttt{description}] = s$.	"Is there a crossing s of/on the intersection?"	If present, answer either with the crossing marking style or "Yes, there is a crossing s." If absent, answer "No, there is no crossing s", sometimes naming another present side.	Existence and spatial perturbation: deny an existing crossing, assert existence at a wrong side, substitute the wrong crossing style, or conflate absence at s with presence elsewhere; optionally include "None of the above".
`lane type`	Select each visible lane l with $\tau(l) =$ `lane`. If an intersection is present, restrict to lanes whose lane-intersection relation is relevant for the ego-intersection distance. Return $(l, p_l[\texttt{type}])$.	"What lane is $d^*(l)$?"	Render the verbose lane class for $p_l[\texttt{type}]$, *e.g.*, standard vehicle lane, turn lane, bike-shared lane, etc.; sometimes answer "None of the above" when the true class is withheld from the options.	Attribute perturbation: replace the true lane class with other classes from the lane-type vocabulary while keeping the selected lane fixed; optional "None of the above".
`crossing type`	Select each visible crossing c near a single intersection with a known $p_c[\texttt{marking_style}]$.	"What is the style of $d^*(c)$?"; "How is $d^*(c)$ marked?"; or "What is the marking of $d^*(c)$?"	Render the pedestrian-crossing style vocabulary value for $p_c[\texttt{marking_style}]$.	Attribute perturbation: replace the true marking style with other pedestrian-crossing style values; optionally include "None of the above".
`traffic light status`	Select each visible traffic light z with $\tau(z) =$ `traffic light`. For vehicle signals, collapse duplicate traffic lights that control the same lane set; Return $(z, p_z[\texttt{status}])$.	"What is the status of $d^*(z)$?"	Render the current status in the last frame as a traffic-light color or as the corresponding pedestrian command.	Attribute perturbation: replace the true status with other legal signal states or pedestrian commands, using the control relation to choose the relevant vocabulary; optionally include "None of the above".

Type	**Selection operator $\mathcal{S}(G_t)$**	**Question template T_q**	**Answer template T_a**	**Graph perturbation / decoy procedure $\mathcal{O}(G_t)$**
`traffic light change`	From $G_{t-w:t}$, select traffic lights z with $\tau(z) =$ `traffic light` where `status` is known for the full window w, $(p_z^0[\texttt{status}], \ldots, p_z^t[\texttt{status}])$.	"Did the status of $d^*(z)$ change over the last w frame(s)?"	If the first and last states differ, render "Yes, it was a first but switched to b." Otherwise render "No, it stayed a." Pedestrian signals use pedestrian-command labels.	Temporal perturbation: create false transition patterns by changing the initial state, final state, or change/no-change claim; also create false constant-status answers and optional "None of the above".
`counting at intersection per direction`	For a single intersection, select lane-intersection relation pairs $(r, \bar{r})$ such as `approaches to the left of` / `leaves to the left of`. Count lanes l with (l, I, r) or $(l, I, \bar{r}) \in E_t$, subject to the ego-intersection distance.	"How many lanes are there $\mathrm{phrase}(r, \bar{r})$?"	Render the total count and directional breakdown, *e.g.*lanes travelling away from versus towards the camera for the selected intersection side or road segment.	Count perturbation: keep the queried relation pair fixed but perturb the two relation-specific counts by nearby integers; optionally replace one option with "None of the above".
`counting per direction`	Select all visible lanes when the scene is off-intersection, or lanes related by `leads up to` / `leaves` when an intersection is far enough away. Count them by $p_l[\texttt{direction}] \in \{\texttt{same}, \texttt{opposite}\}$.	"How many lanes are there?"	Render the total number of lanes and the breakdown into ego-direction and opposite-direction lanes.	Count perturbation: perturb the same/opposite direction count pair by nearby integers while keeping the road extent fixed; optionally include "None of the above".
`counting generic`	For each direction $d \in \{\texttt{same}, \texttt{opposite}\}$, select visible lanes l satisfying $p_l[\texttt{direction}] = d$, restricted to the off-intersection road extent when needed.	"How many lanes are there in ego direction?" or "How many lanes are there in direction opposite to the ego?"	Render a scalar count sentence, *e.g.*"There are k lanes in ego direction" or "There are no lanes travelling towards the camera".	Scalar count perturbation: replace the true count k with nearby nonnegative integers, with one-way constraints for ego-direction counts; optionally include "None of the above".
`pointing`	Select each visible localizable node n which has a defined position in $\mathbb{R}^2$. For lanes and crossings, require another candidate of the same type; for signs, lights, markings, cones, barrels, racks, bollards, and dividers, use candidates from the same broad visual class.	"Where is $d^*(v)$?"	Render the point or bounding box extracted for n's visual localization.	Identity/location perturbation: replace the target marker with markers for visually similar or same-class nodes at different identities or locations.
`counting crossing`	Select all visible crossing nodes c with $\tau(c) =$ `crossing`. If each crossing has a canonical location description, count by location (*e.g.*, at the far side of the intersection, left of the intersection); otherwise count the crossings generically.	"How many pedestrian crossings are there?"	Render the total and, when available, the location layout, *e.g.*crossings to the left/right/straight ahead/far side.	Count/layout perturbation: perturb the total number of crossings or replace the true set of crossing locations with another canonical combination; optionally include "None of the above".
`vehicle position`	Select each actor a (ego or annotated vehicle actor, such as `bus`, `truck`, `emergency vehicle`) with a unique lane l such that $(a, l, \texttt{is in}) \in E_t$ or $(l, a, \texttt{contains}) \in E_t$.	"Which lane is $d^*(a)$ in?"	Render the descriptor $d^*(l)$ of the lane containing the actor.	Relation perturbation: replace the true containing lane with adjacent left/right lanes, lanes containing other actors, or other visible lanes. The true containment edge is hidden during descriptor construction so the answer is not leaked.
`sign controls lane`	Select each sign s with a nonempty controlled lane set $L_s = \{l : (s, l, \texttt{controls}) \in E_t\}$.	"Which lane(s) does $d^*(s)$ control?"	Render the joined descriptors of all lanes in L_s.	Controlled-set perturbation: substitute graph-near lane sets built from adjacent lanes, lanes containing vehicles, and other visible lanes; vary set cardinality as well as identity. Control true edges are hidden during descriptor construction.
`traffic light controls lane`	Select each traffic light z with a nonempty controlled lane set $L_z = \{l : (z, l, \texttt{controls}) \in E_t\}$.	"How many lanes does $d^*(z)$ control?"	Render "$\lvert L_z \rvert$, $d^*(l_1)$ and $d^*(l_2)\ldots$", *i.e.*the number and descriptors of controlled lanes.	Controlled-set perturbation: substitute adjacent or otherwise plausible lane sets, including sets with the wrong cardinality, while preserving road-local plausibility. Control edges are hidden during descriptor construction.

Type	Selection operator $\mathcal{S}(G_t)$	Question template T_q	Answer template T_a	Graph perturbation / decoy procedure $\mathcal{O}(G_t)$
`pairwise vehicle location`	Select unordered pairs of visible actors (a_1, a_2), including ego, where each actor has exactly one containing lane l_1, l_2. Determine whether $l_1 = l_2$ and, if not, whether l_1 is left/right of l_2.	"Are $d^*(a_1)$ and $d^*(a_2)$ in the same lane?"	If $l_1 = l_2$, render "Yes, they are in $d^*(l_1)$." Otherwise render a "No" answer with the two lane descriptors or the left/right relation.	Comparison perturbation: flip the same-lane claim, reverse the left/right relation, or substitute one/both actor lanes with adjacent or common decoy lanes.
`pairwise lane comparison by direction`	Select unordered pairs of visible lanes (l_1, l_2), excluding descriptors or intersection-side relations that would make the comparison trivial. Return p_{l_1}[`direction`] and p_{l_2}[`direction`].	"Do $d^*(l_1)$ and $d^*(l_2)$ share the same direction?"	Render one of four templates: yes, both in ego direction; yes, both opposite ego; no, first ego and second opposite; no, first opposite and second ego.	Comparison perturbation: substitute the incorrect same/different direction templates, including reversed first/second assignments; optionally include "None of the above".

A.2 Construction of chain-of-thought traces

This appendix details how the Chain-of-Thought (CoT) traces are constructed from the CRS graph in the implementation. The generator does not annotate reasoning traces separately. Instead, each trace is derived from the same query instantiation that produces the question, answer, and decoy options.

Let a query be written as

$$q = (S, T_q, T_a, O),$$

where S selects graph elements, T_q renders the question, T_a renders the answer, and O defines perturbations used for hard negatives. For a selected target node n, the generator first samples a unique descriptor $d^\star(n)$. Concretely, each descriptor stores its rendered description text in natural language, the target node, reasoning depth, its uniqueness anchor with respect to property or relation, and a list of dependencies

$$\Delta(d^\star) = \{(u_i, \rho_i, v_i, h_i)\}_{i=1}^{m},$$

where u_i is an intermediate node that must be resolved, ρ_i is the relation used to connect it to the downstream node v_i, and h_i is the hop depth. Descriptors can be obtained from four high-level sources: one of the three uniqueness anchors (unique node type, unique property, recursively unique edge), or a geometric point marker. If no unique descriptor is available, the query is rejected unless the template explicitly allows non-unique enumeration (for example, in counting queries).

The resulting CoT is the concatenation of three graph-derived links:

$$C(q) = C_{\text{anchor}}(q) \circ C_{\text{traverse}}(q) \circ C_{\text{extract}}(q).$$

The exact surface form varies by template, but the underlying computation is shared.

Link 1: Anchor identification. The first link grounds the initial object(s) from which the rest of the reasoning can proceed. For descriptor-based queries, the implementation sorts the dependencies in $\Delta(d^\star)$ by hop depth and begins with the innermost uniquely identifiable node. If the descriptor has no dependencies, the target node itself is the anchor. For a node u, grounding is rendered from its geometric property $p_t^{\text{geo}}(u)$, which becomes a camera-view mention with a grounding point or box.

Operationally, this link emits statements of the form:

$$\begin{aligned}&\texttt{Step 1: Identify the <}d^*(u)\texttt{>, which is visible in the}\\&\texttt{LEFT/CENTER/RIGHT view at <point>}(x, y)\texttt{</point>/}\\&\texttt{<box>}(x_1, y_1, x_2, y_2)\texttt{</box>.}\end{aligned} \tag{5}$$

This identifies the object named by the unique descriptor, then states where it is visible.

Point-based descriptors first identify the marker and then the object marked by it.

$$\begin{aligned}&\texttt{Step 1: Identify the location of <point_1> in the last}\\&\texttt{frame. The marker is visible in the}\\&\texttt{LEFT/CENTER/RIGHT view at <point>}(x, y)\texttt{</point>}\\&\texttt{and marks a <}d(u)\texttt{>.},\end{aligned} \tag{6}$$

with $d(u)$ being a description of the marked object which is not necessarily unique. In counting and existence templates, the selection operator S provides a set of anchors rather than a single descriptor; each counted lane or crossing is introduced as an enumerated object and grounded in the image.

Link 2: Graph traversal. The second link resolves the nested relational part of the descriptor. If a descriptor depends on a chain of nodes

$$u_0 \xrightarrow{\rho_1} u_1 \xrightarrow{\rho_2} \cdots \xrightarrow{\rho_n} n,$$

the trace walks this chain from the already-grounded anchor u_0 to the target n. Each step instantiates the canonical relation statement in language, until reaching the target node, for example

$$\begin{aligned}&\texttt{Step 2: The } \tau(n) \texttt{ in question is the } \tau(n) \texttt{ that <}\rho_n\texttt{>}\\ &\qquad\texttt{that <}\tau(u_{n-1})\texttt{>,and is visible in the LEFT/CENTER/}\\ &\qquad\texttt{RIGHT view at <point>}(x,y)\texttt{</point>/}\\ &\qquad\texttt{<box>}(x_1,y_1,x_2,y_2)\texttt{</box>.}\end{aligned} \tag{7}$$

For templates with more than one relevant object, this traversal is repeated. Comparison templates resolve both compared objects, relation templates resolve both the controller and the controlled lane set, and option-verification templates additionally resolve candidate answer descriptors before comparing their node identities to the target node identities. Counting templates use a related pattern: each selected item is introduced, optionally related to previously introduced lanes through adjacency, and then added to the running enumeration.

Link 3: Target extraction. The final link extracts the answer-bearing quantity from the resolved target node, a linked answer node, or a target set. Before revealing the answer, most templates add a short auxiliary description of the target by stating both its unique and non-unique edges and properties, to anchor it linguistically in the scene.

More concretely, for the resolved target node, the generator forms a local evidence set by selecting up to i properties of the node and up to j outgoing relations to neighboring nodes. For each selected relation, the neighboring node is itself summarized by up to l of its properties. The candidate properties and relations are first randomized and then ordered by a predefined priority hierarchy for the corresponding node type, so the rendered evidence is varied while still preferring semantically useful facts. These selected facts are finally converted into natural-language sentences using the canonical property and relation forms. For $i = j = l = 1$, this yields:

$$\begin{aligned}&\texttt{Step 3: The <}\tau(n)\texttt{>'s <}k^i\texttt{> is <}y^i(t)\texttt{>.}\quad\texttt{The <}\tau(n)\texttt{> <}\ell_{n,n_j}(t)\texttt{>}\\ &\qquad\texttt{<}\tau(n_j)\texttt{> with <}k^l\texttt{> <}y^l(t)\texttt{>.}\end{aligned} \tag{8}$$

The extracted target quantity depends on the query family. For property queries it is a property value, such as lane type, lane direction, line color, crossing marking style, or traffic-light status; for lane-line queries, the answer-bearing node is reached through the lane-marking relation. For relation queries it is a set of target lanes connected by a relation such as `contains` or `controls`. For counting queries it is the cardinality, sometimes decomposed by direction or crossing location. For comparison queries it is an equality or relative-position judgment. For temporal queries it is a sequence of property values across frames. For pointing queries it is the option whose marker corresponds to the resolved target node. The extracted value is rendered through T_a, compared to the answer options, and followed by a conclusion selecting the correct answer.

Table 5 summarizes how different queries used in this study (cf. Table 4) instantiate the three links.

Table 5: CoT construction logic for query families.

Query family	Anchor identification $C_{\text{anchor}}(q)$	Graph traversal $C_{\text{traverse}}(q)$	Target extraction $C_{\text{extract}}(q)$
Property-related (`lane direction`, `lane type`, `line marking`, `line type`, `line color`, `crossing type`, `traffic light status`)	Resolve the descriptor of the queried lane, crossing, lane line, or traffic light and ground it in the camera views.	Follow any nested descriptor dependencies, such as lane adjacency, containment, marking, or control relations, until the queried target node is identified.	Describe auxiliary non-answer properties or edges, then read the requested property from the target node or linked answer node. Traffic-light status additionally checks the latest frame and may cross-verify earlier frames.
Temporal property changes (`traffic light change`)	Resolve and ground the traffic light directly or the lane/crossing controlled by it.	If several lights control the same target, enumerate their visible positions; otherwise follow the descriptor to the unique signal.	Read the status timeline over the frame window and answer whether the value stayed fixed or changed from the first to the last frame.
Relations (`vehicle position`, `sign controls lane`, `traffic light controls lane`)	Identify the actor, sign, or traffic light that appears in the question.	Traverse `is in/contains` or `controls/is controlled by` edges to obtain the target lane or lane set. Candidate options are resolved by their own descriptors.	Compare option node identities with the target lane identities and select the option that describes the correct lane or lane set.
Counting (`counting at intersection per direction`, `counting per direction`, `counting generic`, `counting crossing`)	Set the counting scope, usually the most recent frame and optionally the relevant intersection region, then introduce each selected lane or crossing as a grounded item.	For each item, resolve a short descriptor and, when useful, relate it to the ego lane, an intersection, or previously enumerated neighboring lanes.	Aggregate the enumerated items into totals, optionally split by traffic direction or crossing location, and eliminate/count-match the answer options.
Comparison (`pairwise vehicle location`, `pairwise lane comparison by direction`)	Identify and ground both compared actors or lanes.	Resolve the lane containing each actor, or the descriptor of each lane, then compare the resulting graph nodes or their direction properties.	Return whether the objects share a lane or direction, or state the relative lane position when they differ.
Existence and pointing (`existence of crossings`, `pointing`)	For crossing existence, anchor the requested side of the intersection. For pointing, anchor the object descriptor in the question.	Search the selected crossing set for the requested location, or compare each marker option to the target node.	Answer existence plus marking style when available, or select the marker option whose node is the target.

A.3 Recursive construction of unique node descriptors

This appendix describes the descriptor recursion used to turn the uniqueness signals in the CRS graph into natural-language object references. For a target node n, the procedure returns a set $D(n)$ of candidate descriptors $d(n)$. Each candidate stores its rendered text, the target node, whether the reference is unique, the number of relational hops, and the intermediate nodes that must be resolved when the descriptor is later verbalized as a reasoning trace.

Let $p_t(n)$ denote the property dictionary of n, let $\tau(n)$ denote its node type, and let $U(n)$ be the list of property keys marked as unique for n. The implementation also precomputes an edge anchor map from all edges E_t and the list of edge labels marked as unique, $U(\ell_{n,n'}(t))$ as

$$A(n) = \{(m, \ell_{n,m}(t)) : (n, m) \in N_t,\ l_{n,m}(t) \text{ is marked unique}\},$$

where m is the neighboring anchor node and $\ell_{n,m}(t)$ is the edge label. In our implementation, all unique edge labels are directionally applied to outgoing edges from the target node.

The construction is a candidate-generating cascade rather than an early-exit priority list. Node-level uniqueness, unique properties, recursive unique-edge descriptors, and point-marker fallbacks are all appended when available. Only when no such descriptor can be built does our algorithm return a non-unique label. Recursive edge descriptors are bounded by a hop budget H (equivalent to the generated maximum reasoning depth) and a visited set to prevent cycles.

Property rendering. Unique properties are treated differently depending on their key. If the key is `description`, the property value is assumed to already be a compact visual noun phrase, such as "blue sedan" or "crossing to the right". The descriptor is therefore rendered directly as

$$\text{"the } p_t(n)[\texttt{description}]\text{"}.$$

Algorithm 1: Recursive construction of node descriptors.

Input: frame graph G_t, target node n, unique-edge anchor map A, hop budget H, visited set V
Output: candidate descriptor set $D(n)$
Function `BuildDescriptor`(G_t, n, A, H, V)**:**

- **if** $n \in V$ **then**
  - **return** $\emptyset$
- $V \leftarrow V \cup \{n\}$
- $D \leftarrow \emptyset$
- $P \leftarrow p_t(n)$, $\nu \leftarrow \tau(n)$
- **if** $P[$*is_unique*$]$ **then**
  - add $\langle$"the ν", n, unique, $0, \emptyset\rangle$ to D
- **foreach** $k \in P[$*unique_properties*$]$ **do**
  - **if** $k =$ *description* **then**
    - $s \leftarrow$ "the $P[k]$"
  - **else**
    - $s \leftarrow$ "the ν whose k is $P[k]$"
  - add $\langle s, n, \text{unique}, 0, \emptyset, \text{property} = k\rangle$ to D
- **if** $H > 0$ **then**
  - **foreach** $(m, \ell_{n,m}(t)) \in A(n)$ **do**
    - $C \leftarrow$ `BuildDescriptor`$(G_t, m, A, H-1, V)$
    - **foreach** $c \in C$ **do**
      - $s \leftarrow$ "the ν that ρ c.text"
      - $\Delta \leftarrow c.\text{deps} \cup \{(m, \ell_{n,m}(t), n, c.\text{hops})\}$
      - add $\langle s, n, c.\text{unique}, c.\text{hops}+1, \Delta, \text{relation} = (\ell_{n,m}(t), c.n)\rangle$ to D
- **if** $P[$*position*$]$ *exists* **then**
  - add $\langle$"the ν at $\langle P[$`position`$]\rangle$", n, unique, $0, \emptyset\rangle$ to D
- **if** $D = \emptyset$ **then**
  - **return** $\{\langle n, n, \text{nonunique}, 0, \emptyset\rangle\}$
- **return** D

For all other unique properties, the node type remains explicit and the property is rendered as a constraint:

$$\text{"the } \tau(n) \text{ whose } k \text{ is } p_t(n)[k]\text{"}.$$

This distinction avoids awkward phrases such as "the vehicle whose description is blue sedan" while preserving explicit property grounding for keys like lane direction, lane type, line color, or traffic-light status.

The recursive case transfers uniqueness from an already unique neighboring descriptor to the target node through a unique relation. For example, if a lane can be uniquely described as "the ego lane", then a neighboring lane connected by a unique `left of` relation can be described as "the lane that is left of the ego lane". The dependency list records the intermediate node and relation so that the later CoT construction can first ground the anchor and then walk the relation chain back to the queried target. The algorithm is detailed in Algorithm 1, while an example of substituting unique descriptors into a query is provided in Figure 8.

A.4 Semi-automated graph construction with human enrichment

This appendix describes the data-ingestion pipeline used to initialize the graph interface. The goal of this preprocessing is not to produce a final scene graph automatically. Instead, it converts several heterogeneous autonomous-driving annotations into a common, frame-indexed representation that can be displayed in the annotation tool and then corrected, selected, and enriched by human annotators.

Input sources. For each Argoverse2 Sensor [21] log, we use the front-left, front-center, and front-right camera streams, ego poses, camera calibration, tracked 3D object annotations, and the

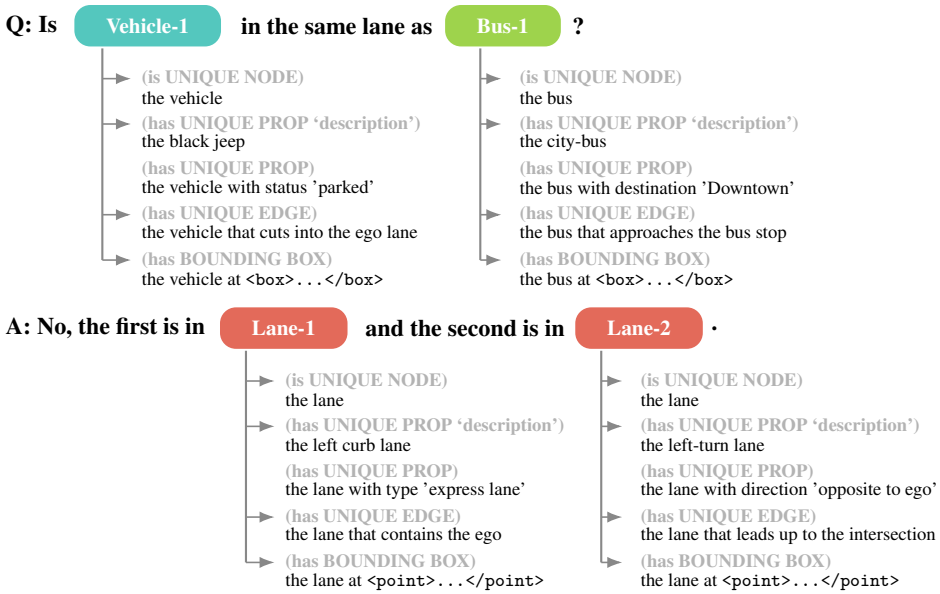


Figure 8: Substituting all potential unique descriptors in relatively simple question templates leads to complicated multi-hop reasoning tasks and broad linguistic variety.

Argoverse HD map. We additionally use OpenLane-V2 [4] annotations for traffic lights, road signs, their image-space boxes, and their topological association to lane segments. The derived annotations contain scene metadata such as image identifiers, per-camera calibration, and ego poses, as well as a broad range of scene elements (`lane_segments`, `lane_lines`, `intersections`, `splits`, `merges`, `pedestrian_crossings`, `traffic_elements`, and `objects`). Dynamic quantities are keyed by an integer frame index.

Map and traffic-control ingestion. We first aggregate OpenLane-V2 frames that refer to the same Argoverse source log. OpenLane-V2 traffic elements are merged over time by their element ID: traffic lights keep a per-frame state in `status`, while road signs keep a symbolic `meaning`.

We then load the corresponding Argoverse HD map. Lane topology is made bidirectionally consistent, geometrically contiguous lane segments are merged when their markings and topology permit it, and lane-marking boundaries are also merged into explicit lane-line objects. For every lane, we store its `lane type` $\in$ [`vehicle`, `bus`, `bike`]. For every lane line we store its 3D polyline, `marking type`, as well as geometric connectivity to neighboring line segments and the lanes for which it is the left or right boundary, for semi-automatic edge instantiation.

Intersection regions are formed by clustering HD-map lane polygons marked as intersection segments. Each cluster stores a 3D boundary, incoming and outgoing lanes, overlapping pedestrian crossings, and turn relations between incoming and outgoing lanes (for semi-automatic edge creation). Lane splits and lane merges outside of intersections are detected from successor/predecessor branching in the lane topology. Finally, map elements are filtered by visibility: at each ego pose we construct a forward-facing local window and keep only lanes, lane lines, crossings, intersections, splits, and merges that intersect that window.

Object ingestion. Tracked Argoverse objects are synchronized to each camera frame by selecting the closest 3D detection timestamp. Because camera frames and lidar/object timestamps are not identical, lidar points and object poses are warped through the ego-pose trajectory into the camera-frame timestamp. We project each 3D box into the front cameras using the camera calibration, and keep boxes within the image and within the configured distance range of 50m. The output object entry stores the object type, as well as per-camera and per-frame 2D boxes. Additionally, we algorithmically determine the position of objects in the scene by testing if they are contained in one of the derived lane or intersection polygons.

Human enrichment in the graph tool. The graph creation tool in Figure 9 displays the preprocessed scaffold as overlays on the three front camera images and as a top-down map view. Human annotators then convert the scaffold into the final graph by selectively transferring relevant objects. Clicking a projected source element creates a typed graph node with a unique node ID by transferring the pre-processed properties. Annotators can also add manual point or bounding-box nodes when the source data misses an element, and they can attach or delete camera-specific position markers for a node in a specific frame. Each node has a type, properties, helper metadata such as position in the image space, and optional uniqueness anchors. Properties are template-driven but editable; 'locked' properties are treated as static, while 'unlocked' properties are stored per frame and can be propagated forward or backward in time.

Edges are created both automatically and manually. Automatic edges use the source-derived helper fields: objects are dynamically placed in lanes or intersections ("is in"), lanes connected to traffic lights and those signs that are part of OpenLane-V2 ("controls"), lane neighbors are connected through pre-processed HD map data ("is left / right of"), as well as lane lines and their respective lanes ("is left / right marking of"), crossings to their intersection ("is on"). In the graph editor, annotators can then add arbitrary edges, choose whether the predicate is static or frame-varying, and edit the label in natural language. They can also remove edges for the current frame or globally, and propagate temporal edge labels across neighboring frames. The exported annotation therefore contains the refined source-derived geometric scaffold plus the human-curated semantic graph: selected nodes, corrected temporal attributes, natural-language predicates, and the uniqueness anchors used to identify objects in downstream question generation. An overview of the components of this process is provided in Table 6.

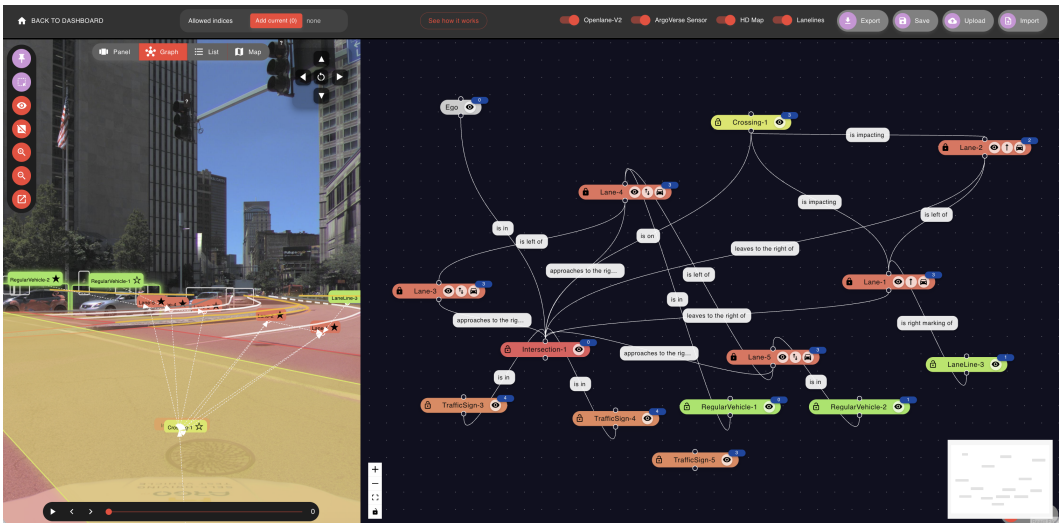


Figure 9: The graph creation tool semi-automatically creates a spatio-temporal scene graph, while human annotators filter and enrich the graph.

Table 6: Source-derived fields used to initialize the graph annotation tool and the corresponding human enrichment operations. The automatic fields provide geometry, topology, visibility, and coarse semantics; human annotation converts this scaffold into the final graph by validating elements, correcting temporal properties, and adding natural-language node and edge semantics.

Element	Primary source	Automatic representation	Human correction and enrichment
Image sequence and ego state	Argoverse camera streams, ego poses, and calibration	-	Used as visual context for all annotation decisions.

Element	Primary source	Automatic representation	Human correction and enrichment
Lanes	Argoverse HD map lane polygons and topology, with OpenLane-V2 traffic-control topology	`lane` nodes with left/right boundary geometry, coarse `lane type` and per-frame visibility. Semi-automated edge annotations are stored from left/right neighbor IDs.	Annotators instantiate only relevant lanes, add properties such as `direction` or `variant = one_way`, remove false or irrelevant lanes, add or correct lane adjacency relations, and connect lanes leaving or approaching intersections.
Lane lines	Argoverse HD map lane boundaries	`lane_lines` with 3D geometry, `mark_type` and per-frame visibility. Semi-automated edges are stored from `left_of/right_of` lane references.	Annotators confirm which projected markings should enter the graph, correct marking properties, and express or correct marking relations such as "is left marking of" or "is right marking of".
Traffic lights, road markings and signs	OpenLane-V2 traffic elements and topology matrix	`traffic_lights` and `signs` with category, per-frame image bounding boxes, traffic-light `status`, or sign `meaning`; semi-automated lane associations are stored for edge creation.	Annotators decide whether each element should become a graph node, correct traffic-light `state` and `type` (traffic lights for main road vs pedestrian or bike signals), refine sign meaning, and edit or delete regulatory edges. Additionally, human oversight is needed for differentiating between road markings and signs, which are not treated as separate categories in OpenLane-V2.
Pedestrian crossings	Argoverse HD map pedestrian crossing polygons	`pedestrian_crossings` with geometry and visibility. Semi-automatic edges are created between crossings and intersections ("is in").	Annotators confirm crossings that are semantically relevant and connect them to intersections with natural-language predicates such as "is on", define additional properties such as `style` or `marking_quality`, among others.
Intersections and turn structure	Argoverse HD map lane polygons and predecessor/successor topology	`intersections` with geometry, incoming and outgoing lanes, linked crossings.	Annotators select visible intersections, validate incoming/outgoing structure, and edit relations that describe approach, departure, and allowed maneuvers, as these can not reliably be extracted programmatically in natural language for complex intersections.
Lane splits and merges	Derived from lane predecessor/successor branching	`splits` and `merges` store a 3D event point, incoming/outgoing lanes, and per-frame visibility.	Annotators instantiate or discard these event nodes and edit the topology predicates used in the final graph.
Tracked objects	Argoverse 3D tracked object annotations	`objects` with object type, per-camera per-frame 2D boxes, and per-frame lane/intersection relationship ("is in").	Annotators choose which objects matter, correct object class or visibility, add missing image positions with point/box clicks, and refine object descriptions, properties, etc.
All remaining nodes	human annotation	-	Annotators add, delete, or rewrite edge labels and node properties in natural language for interesting, missing or relevant objects in the scene. This can be everything within and beyond the ingested annotations.

Annotation effort and quality control. Average annotation time is approximately 5 minutes per CRS scene graph, though it varies considerably across scenes: annotation is a selective curation process, not exhaustive labeling: annotators include only the entities, attributes, and relations they judge salient for a given scene, so time scales with what an annotator chooses to curate, not with a fixed per-scene inventory. On quality control, we deliberately do not report a classical inter-annotator agreement statistic (e.g., label overlap or Cohen's kappa), because such metrics assume a single ground-truth annotation two annotators should converge on, which does not hold here by design: CRS operates under an explicit open-world, no-completeness assumption (Section 3.3) — a graph need not include every instance of an entity type unless completeness is required for a specific query — so two annotators can validly produce graphs with different emphasis (different nodes included, different attributes added, different uniqueness anchors chosen) without either being incorrect.

What quality control does target, and what actually matters for validity, is not agreement on what was selected but correctness of what was included: is a given property value accurate, is a relation correctly labeled, is a chosen descriptor actually unique. We check this via (i) a second-pass review

of every annotated graph by a second person, verifying well-definedness and correctness rather than agreement with the first annotator's selections, and (ii) manual verification of the generated question-answer pairs specifically for the held-out validation set (our existing expert-verified set, see Section 4).

A.5 CRS-enriched scene statistics

This appendix reports the graph statistics for the 80 CRS-enriched scenes used in this study. Following the notation in Section 3, each annotated scene is treated as a temporal directed graph representation $G = \{G_0, \ldots, G_T\}$, with per-frame graphs $G_t = (N_t, E_t)$ with a temporal window size of $w = 4$. For each training example, the first $w - 1$ frames provide temporal context, and statistics are accumulated only for the queried current frame. Thus, the 80 CRS-enriched scenes for training yield

$$M = \sum_{G} G_t = \sum_{G} |\{t : t \geq w - 1\}| = 2{,}320$$

evaluated frame graphs, with an average of 29.0 queried frame graphs per CRS graph.

The number of CRS graphs is small at the scene level but dense at the graph-observation level. An average queried frame graph contains 14.79 nodes, 38.46 directed edges, and 87.12 node-property entries. Equivalently, nodes have an average incoming-or-outgoing edge incidence of $2|E_t|/|N_t| = 5.20$ and carry 5.89 properties. The uniqueness signal used by recursive descriptors is also present throughout the train graphs: an average frame graph contains 1.63 uniquely identifiable node anchors, 16.76 unique edge anchors, and 3.75 unique property anchors. Completeness is more frequent for lanes than for crossings: $1{,}904$ frame graphs are lane-complete (82.1%), while 932 are crossing-complete (40.2%). Table 7 summarizes these findings.

Table 7: CRS graph statistics. Totals are accumulated over the $M = 2{,}320$ evaluated frame graphs induced by the 80 CRS graphs used in training.

Statistic	**Total**	**Mean**	**Unit**
Evaluated frame graphs $\lvert \sum G(t) \rvert$	2,320	29.00	per CRS graph
Node observations $\sum_t \lvert N_t \rvert$	34,316	14.79	per G_t
Directed edge observations $\sum_t \lvert E_t \rvert$	89,230	38.46	per G_t
Node-property entries $\sum_{t,n} \lvert P_t(n) \rvert$	202,117	87.12	per G_t
Incoming/outgoing edge incidence $2\lvert E_t \rvert / \lvert N_t \rvert$	–	5.20	per node
Node-property density	–	5.89	per node
Lane-complete frame graphs, $c_t(\text{lane}) = 1$	1,904	82.1	percent of G_t
Crossing-complete frame graphs, $c_t(\text{crossing}) = 1$	932	40.2	percent of G_t
Unique node anchors	3,785	1.63	per G_t
Unique edge anchors	38,891	16.76	per G_t
Unique property anchors	8,702	3.75	per G_t

A.6 Question type analysis

The training set contains 23,343 samples and the validation set contains 1,014 expert-verified samples, both spanning the same 19 question types. These types cover direct perception of road attributes, counting over lanes and crossings, pairwise comparisons, localization/existence queries, and relational control or containment queries. We assign each type to the same reasoning split and evaluation bucket used in the benchmark analysis: *perception-like* questions primarily ask for visible attributes which require lower degrees of reasoning, counts, temporal status changes, or simple pairwise lane-direction comparisons, while *reasoning-heavy* questions require resolving a localized target or traversing a graph relation before selecting the answer.

A.7 Evaluation protocol

All open- and closed-source baselines, as well as our finetuned models, were prompted with the identical, single-turn, zero-shot prompt template with explicit CoT instructions shown in full below,

Table 8: Question type distribution in the training and validation sets.

Question type	Train	Val.	Asks about	Reasoning	Bucket
`counting at intersection per direction`	1,388	67	Lane counts by approach/exit direction at an intersection	Low	Counting
`counting crossing`	225	43	Number and layout of pedestrian crossings	Low	Counting
`counting per direction`	1,087	74	Lane counts split by ego/opposite direction off intersection	Low	Counting
`counting generic`	2,174	80	Generic lane count for a requested direction or road extent	Low	Counting
`lane direction`	2,185	75	Whether a lane follows or opposes the ego direction	Low	Properties
`lane type`	1,811	70	Semantic lane class such as vehicle, bike, bus lane	Low	Properties
`line color`	1,150	63	Color of a lane boundary marking	Low	Properties
`traffic light status`	1,546	56	Current traffic-light or pedestrian-signal state	Low	Properties
`line marking`	1,150	57	Full lane-line marking style	Low	Properties
`line type`	1,150	56	Lane-line pattern or visibility type	Low	Properties
`crossing type`	358	33	Pedestrian-crossing marking style	Low	Properties
`traffic light change`	1,673	60	Whether a traffic light changed over the temporal window	Low	Properties
`pairwise lane comparison by direction`	1,339	80	Whether two lanes share the same travel direction	Low	Comparison
`pairwise vehicle location`	1,071	34	Whether two actors share a lane or their relative lane position	High	Comparison
`pointing`	1,656	38	Image location of a referenced scene element	High	Existence
`existence of crossings`	436	43	Presence, location, or style of an intersection crossing	High	Existence
`sign controls lane`	310	14	Which lane set a road sign controls	High	Relational
`traffic light controls lane`	1,345	28	Which or how many lanes a traffic light controls	High	Relational
`vehicle position`	1,289	43	Lane containing the ego vehicle or another actor	High	Relational

with no per-model prompt tuning. References to objects ("the vehicle marked with box 1") were drawn onto the respective images.

VLM Prompt

You are given images captured from an ego vehicle.
Images are provided for the past 4 frames starting from the oldest to the most recent, with 3 images per frame for a total of 12 images.
The question always refers to the last frame.
Images per frame are provided in this exact order:
1. LEFT camera
2. CENTER camera
3. RIGHT camera

Output format rules:
Before providing the final answer, perform a step-by-step visual analysis of the scene, referencing specific cameras or grounding boxes where relevant to justify your conclusion.

Question:
<question text>

Answers:
A. <option>
B. <option>
C. <option>
D. <option>

Table 9: Accuracy is reported as decimals with 95% scene-clustered CI. This table is an extension of Table 1 in the main paper.

Model	Total	Perception-like	Reasoning-heavy
gemma-4-E2B-it [60]	0.384 (0.350, 0.414)	0.415 (0.371, 0.455)	0.260 (0.199, 0.332)
qwen3-vl-30b-a3b-instruct [59]	0.460 (0.424, 0.497)	0.478 (0.439, 0.517)	0.385 (0.322, 0.452)
qwen3-vl-8b-instruct [59]	0.494 (0.462, 0.526)	0.509 (0.473, 0.545)	0.435 (0.358, 0.513)
gemma-4-26b-a4b-it [60]	0.501 (0.470, 0.533)	0.529 (0.494, 0.564)	0.385 (0.320, 0.448)

Table 10: Accuracy over different reasoning depths is reported as decimals with 95% scene-clustered CI.

Model	1-hop	2-hop	3-hop	>3-hop	-$\Delta_{1\to4}$
GPT-4o-mini [57]	0.49 (0.40, 0.57)	0.53 (0.44, 0.61)	0.52 (0.43, 0.61)	0.38 (0.29, 0.47)	21.5%
Gemini-3.1-Flash-Lite [17]	0.67 (0.60, 0.74)	0.64 (0.57, 0.71)	0.53 (0.43, 0.63)	0.43 (0.35, 0.51)	36.3%
Claude-Sonnet-4.5 [19]	0.61 (0.54, 0.68)	0.57 (0.49, 0.65)	0.57 (0.46, 0.67)	0.51 (0.43, 0.59)	16.3%
Gemini-3.1-Pro [17]	0.75 (0.69, 0.82)	0.70 (0.61, 0.78)	0.50 (0.39, 0.60)	0.53 (0.44, 0.61)	30.2%
GPT-5.4 [18]	0.77 (0.70, 0.84)	0.76 (0.68, 0.84)	0.65 (0.56, 0.74)	0.50 (0.42, 0.57)	35.8%
Gemma-4-E2B-IT [60]	0.44 (0.37, 0.53)	0.41 (0.35, 0.48)	0.40 (0.29, 0.50)	0.27 (0.20, 0.35)	37.8%
Molmo2-8B [11]	0.52 (0.45, 0.60)	0.42 (0.33, 0.49)	0.41 (0.31, 0.50)	0.33 (0.23, 0.42)	37.3%
Kimi-k2.6 [58]	0.60 (0.54, 0.65)	0.49 (0.40, 0.58)	0.38 (0.29, 0.46)	0.37 (0.29, 0.46)	37.4%
Qwen3-VL-30b-a3b-Instruct [59]	0.54 (0.47, 0.62)	0.57 (0.48, 0.65)	0.50 (0.42, 0.58)	0.39 (0.30, 0.48)	28.1%
Qwen3-VL-30b-a3b-Thinking [59]	0.53 (0.45, 0.61)	0.47 (0.39, 0.56)	0.52 (0.42, 0.63)	0.46 (0.38, 0.53)	14.0%
Qwen3-VL-235b-a22b-Instruct [59]	0.66 (0.58, 0.73)	0.61 (0.52, 0.69)	0.50 (0.40, 0.59)	0.48 (0.41, 0.56)	26.8%
Gemma-4-26b-a4b-IT [60]	0.66 (0.59, 0.73)	0.61 (0.51, 0.70)	0.53 (0.44, 0.63)	0.43 (0.35, 0.51)	35.0%
Gemma-4-31b-IT [60]	0.68 (0.60, 0.75)	0.64 (0.54, 0.72)	0.61 (0.52, 0.70)	0.53 (0.44, 0.61)	22.6%
Qwen3-VL-8b-Instruct [59]	0.61 (0.53, 0.68)	0.56 (0.48, 0.64)	0.52 (0.41, 0.63)	0.41 (0.32, 0.50)	32.0%
Qwen3-VL-2b-Instruct [59]	0.53 (0.45, 0.61)	0.49 (0.40, 0.58)	0.42 (0.31, 0.51)	0.31 (0.23, 0.39)	41.2%
+ *supervised fine-tune*	0.82 (0.76, 0.87)	0.75 (0.68, 0.82)	0.72 (0.63, 0.82)	0.68 (0.60, 0.75)	17.1%
+ *supervised fine-tune* + *CoT*	0.80 (0.74, 0.86)	0.76 (0.67, 0.83)	0.73 (0.64, 0.83)	0.69 (0.62, 0.75)	14.2%
Qwen3-VL-4b-Instruct [59]	0.55 (0.46, 0.63)	0.53 (0.45, 0.62)	0.47 (0.38, 0.56)	0.36 (0.28, 0.44)	34.3%
+ *supervised fine-tune*	0.81 (0.75, 0.88)	0.72 (0.62, 0.80)	0.81 (0.72, 0.88)	0.70 (0.62, 0.78)	13.8%
+ *supervised fine-tune* + *CoT*	0.80 (0.74, 0.86)	0.76 (0.68, 0.83)	0.78 (0.67, 0.87)	0.71 (0.63, 0.78)	11.4%

We confirmed the model reasoning by inspecting the saved responses, which all consistently contain a multi-step visual analysis before choosing and returning the final answer choice. To assure that models are not unfairly penalized for not returning the exact answer format, we allow flexible returns ("The answer is X", "X", "X: There are two lanes", "There are two lanes", etc...) and randomly inspect answers labeled as wrong manually. We did not include few-shot instructions in this work, for a practical reason worth noting: each example in this benchmark is itself 12 images (4 temporal frames × 3 cameras), so even a 2- or 3-shot prompt multiplies the visual context by a factor 3 or 4, approaching the per-request image-count/context limits.

A.8 Additional quantitative results

Overall accuracy. This subsection details the performance of additional open-source models evaluated on our validation split in addition to those in Table 1. The results are reported in Table 9.

Performance with reasoning depth. This subsection details the performance decrease with reasoning depth (see Section 4 and Figure 5) for all evaluated models. The results are reported in Table 10. Our finetuned models display the smallest delta between performance on queries with lower reasoning depth and queries with higher reasoning depth.

Impact of decoys on no-image baseline. This subsection further analyzes the no-image baseline, corresponding to the result of supervised finetuning (− images) in Table 1. In order to isolate the effects of the linguistic reasoning component from the grounded visual reasoning, we baselined against a model that is fine-tuned with a single image taken randomly from the training dataset, applied across all samples. This renders the model "blind" to the true scene corresponding to the visual question, and is trained to answer based on text cues alone. We report results of this model under the multiple choice setting in Section 4. To de-correlate the visual reasoning from text cues, we train a pair of *free-form* question generation models, using image inputs *vs.* the placeholder fake image input, respectively.

As shown in Figure 10, the performance difference between using visual context (`with image`) is more pronounced than without visual context (`no image`). In order to compute accuracy from free-form answer generation, we limit the question types to 11 that can be matched *via* regex. We further analyze the distribution of the outputs with and without images under the counting task in Figure 11, showing that models that are trained without using images tend to collapse to the most likely answer. This highlights the fact that models trained without visual information can retain road structure, but there is a need for visual grounding for accurate road understanding.

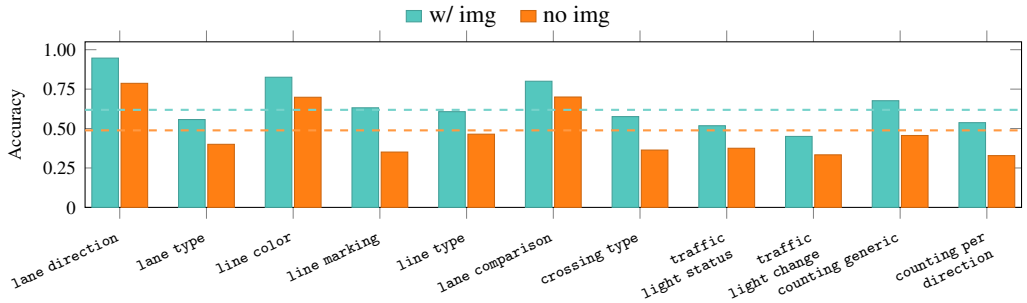


Figure 10: Per-type accuracy for free-form generation comparing models `with image` *vs.* models with `no image`.

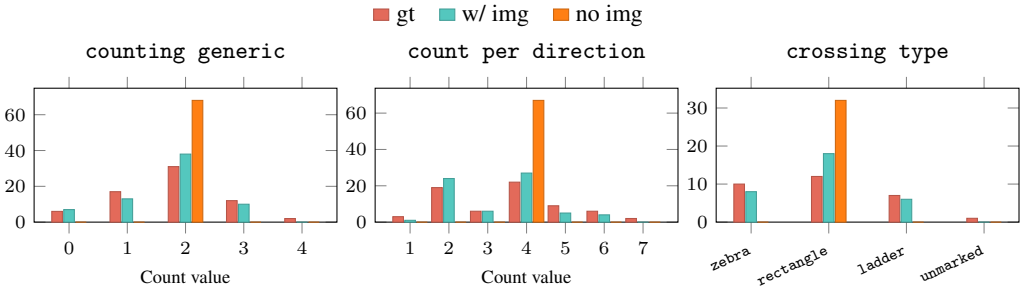


Figure 11: Prediction frequency distributions for 3 descriptive question types.

A.9 Additional qualitative results

We provide additional qualitative results of the output of the finetuned Qwen-4B model trained with CoT reasoning (last row in Table 1) in Figure 12. The reasoning trace is an excellent means of investigating model failures.



QUESTION
What is the left lane line type of the lane that is right of the ego lane?

ANSWERS

A There is no lane line visible.

✓ dashed.

C double dashed.

D double solid.

Reasoning Output:

Step 1: Identify the ego lane, which is visible in the CENTER view at P1.

Step 2: The lane in question is the lane that is right of that lane, and is visible in the CENTER and RIGHT view at P2.

Step 3: The lane's description is rightmost lane. The lane contains a box_truck located in CENTER and RIGHT view at P3.

Step 4: Finally, the lane marking in question is located in the CENTER and RIGHT view at P3. It is DASHED_WHITE.

Conclusion: the correct answer is 'dashed'.

(a) Example output of the model trained with CoT on a question inquiring about the marking type of a lane line.



QUESTION
Is there a crossing to the right of the intersection?

ANSWERS

A Yes, but it is unmarked.

✓ Yes, marked as a zebra crossing.

C No, but there is one to the left.

x None of the above.

Reasoning Output:

Step 1: I am tasked to identify if there is a crossing to the right of the intersection, and determine its potential marking style.

Step 2: Analyzing the sequence of frames and their camera views. I detect a crossing to the right of the intersection, visible in CENTER and RIGHT view at P1.

Step 3: Determining the marking style. This crosswalk is a white outlined rectangle crossing.

Conclusion: Therefore, the correct answer is option 'None of the above'.

(b) Example output of the model trained with CoT on a question requiring existence of a crossing. The model correctly identifies the crossing but fails to extract the correct marking style.



QUESTION

How many lanes are there?

ANSWERS

✓ There are 2 lanes in direction opposite to the ego and 2 lanes leading away from the camera.

B There are 3 lanes travelling towards the camera and 1 lane in ego direction.

C There are 2 lanes travelling towards the camera and 1 lane in ego direction.

D None of the above.

Reasoning Output:

Step 1: To determine the total number of lanes on this road, I categorize the lanes by their traffic direction.

Step 2: I start from the ego lane, the right curb lane, which I call 'Lane 1'. Considering the images, the lane is visible in the CENTER view at P1. The lane's type is VEHICLE. The lane contains a bus with description red bus located in CENTER view at P2. Traffic in this lane moves away from the camera.

Step 3: To the left of Lane 1, I detect another lane, the 2nd lane from the right, which I call 'Lane 2'. Considering the images, the lane is visible in the LEFT and CENTER view P3. The lane's type is VEHICLE. The lane contains a regular_vehicle with description silver sedan and status driving located in LEFT and CENTER view at P4. Traffic in this lane moves away from the camera.

Step 4: To the left of Lane 2, I detect another lane, the 2nd lane from the left, which I call 'Lane 3'. Considering the images, the lane is visible in the LEFT view at P5. The lane's type is VEHICLE. The lane leaves an intersection. Traffic in this lane moves towards the camera.

Step 5: To the left of Lane 3, I detect another lane, the left curb lane, which I call 'Lane 4'. Considering the images, the lane is visible in the LEFT view at P6. The lane's type is VEHICLE. The lane leaves an intersection. Traffic in this lane moves towards the camera.

Step 6: Based on these observations, I count exactly 4 lanes. Out of these, 2 travel away from the camera and 2 travel towards the camera.

Conclusion: Therefore, the correct answer is option A: 'There are 2 lanes in direction opposite to the ego and 2 lanes leading away from the camera.'.

(c) Example output of the model trained with CoT on a counting question. The model steps through all lanes and identifies relevant properties and relations for grounding.

Figure 12: Qualitative examples of the model output when tasked with road reasoning.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in the abstract are elaborated in the end of the Introduction section. All 3 contribution claims are backed by descriptions in Section 3 and experimental results in Section 4. Potential impacts of the claims are described in the Discussion section.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations and potential future work are described in the "Limitations and future work" section in Section 5. All assumptions and robustness details are described in Section 4, with further details in the Supplementary.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: This work does not present a core theoretical contribution. All formulas and equations are numbered and cross referenced.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The work details all information needed to reproduce the main experimental results of the paper in Section 4, as well as additional implementation details in the Supplementary material's Section A.1, Section A.2, Section A.3, and Section A.4. All code and annotated data will be released upon acceptance, including the tool used to implement the CRS framework that is described in this work.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: All data and code will be released and linked in the project page, along with the tooling for implementing the CRS framework as described in the work. Raw sensor data is from the Argoverse2 dataset [21], and is publicly accessible.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All experimental settings and details are included in the Section 4 section, including the graph construction, models, compute, and hyperparameters. All model tuning and analysis details are provided. Additional analysis and experimental details can be found in the Supplementary material, *i.e.*, query construction in Section A.1, graph generation in Section A.4, and statistics analysis in Section A.5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: Statistical significance and error bars are reported in the experiments. Main results in Tab. 1 report the 95% confidence interval along with the first moment. All subsequent experimental results with reported error bars are computed across multiple bootstrapped trials (Section 4).

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The work details compute resources used for this paper at the beginning of Section 4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This work abides by the NeurIPS Code of Ethics, and discusses limitations in the "Limitations and future work" part of Section 5. Data used in this work is anonymized from the provider side (see [21]).

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Potential impact of this work is discussed in Section 5.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This work does not present any scraped datasets, and the method itself poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

Justification: All code packages, base models, and datasets are properly cited and credited.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: Code and tooling to implement the CRS framework will be released, and is well documented in the Supplementary material (Section A). Detailed documentation, including a `README`, will be provided with the code release.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [Yes]

Justification: Details of instructions given to participants are detailed in Section 4 and in the Supplementary material. All expert participants for the small-set human evaluation abide by the NeurIPS Code of Ethics.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [Yes]

Justification: There were no risks presented by the expert survey. Procedures were followed, including the NeurIPS Code of Ethics guidelines and from the authors' institution.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development does not involve LLMs.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.