Edge-Ready 3D Scene-Language Reasoning with Compact Object Graphs

Yatharth Agarwal[1], Arghadip Das[1], Arnab Raha[2], Vijay Raghunathan[1]
[1]Purdue University, West Lafayette, IN, USA [2]Intel Corporation, USA
{agarw414, das169, vr}@purdue.edu, arnab.raha@intel.com

***Abstract*—The proliferation of assistive robotics and Artificial Intelligence of Things demands indoor spatial reasoning capabilities that are semantically rich yet bounded by the latency and energy envelopes of edge devices. Addressing the fundamental conflict between the complexity of spatial reasoning and these resource constraints, we present an edge-ready 3D scene–language pipeline that decouples perception from reasoning through a compact, object-centric relational representation. Rather than relying on computationally expensive dense geometry or 2D feature extraction, we distill instance-segmented scenes into a sparse graph of object identities and local spatial relations, explicitly designed for efficient LLM prompting. This representation is serialized into a deterministic prefix, enabling a small, 4B parameter LLM to resolve complex spatial queries in a zero-shot manner without task-specific fine-tuning. By constraining both input structure and output schema, we bound prompt length and decoding cost while maintaining sufficient relational context for compositional indoor language. To enable practical deployment on edge hardware, we introduce a holistic end-to-end system design: we eliminate host-side bottlenecks via a fully GPU-resident superpoint partitioning routine and improve inference efficiency through prefix KV-cache reuse and low-precision FP4 quantization. Evaluated on an NVIDIA Jetson platform, our system achieves a 3.2× end-to-end speedup and a 1.6× improvement in energy efficiency. At the same time, it delivers state-of-the-art zero-shot performance, surpassing strong prior methods by +15.95% Acc@0.50 on complex 3D reasoning segmentation and by +2.6% Acc@0.50 on ScanRefer. Our results demonstrate that compact object-level relations provide an effective and efficient substrate for spatial reasoning, enabling high-fidelity 3D scene understanding within edge device resource budgets.**

***Index Terms*—3D scene understanding, vision–language reasoning, edge inference optimization**

I. Introduction

The proliferation of Artificial Intelligence of Things (AIoT) has created an urgent need to push Deep Learning (DL) from the cloud to the network edge. Although edge intelligence promises interactive AI with reduced latency and enhanced privacy, it faces a key challenge: the increasing complexity of DL models, particularly in vision–language tasks, conflicts with the strict performance, memory, and power envelopes of edge devices. This tension is most acute in indoor environments, such as assistive robotics and safety monitoring, where systems must process high-volume sensor data (*e.g.*, RGB-D streams) to support complex closed-loop interactions.

In these settings, recognition alone is insufficient; applications require spatial reasoning, the ability to understand structural relationships, such as identifying *"the chair left of the desk,"* under strict latency constraints. Conventional approaches often fail to meet these requirements: cloud offloading introduces unacceptable latency variability, while on-device deployments of state-of-the-art 3D vision–language models (VLMs) remain bottlenecked by heavy perception pipelines and large intermediate representations [1].

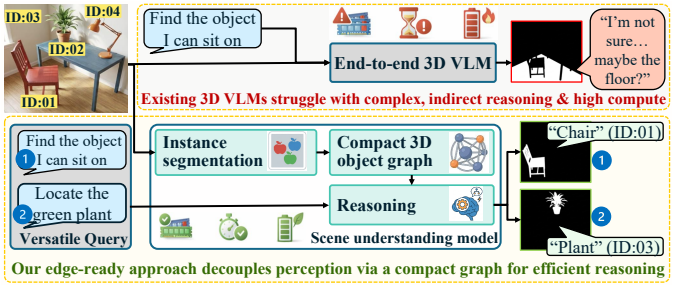

Fig. 1. End-to-end pipeline: Instance segmentation yields object instances; a compact object-relation graph is serialized as text and queried by an instruction-tuned LLM with schema-constrained outputs.

To resolve the conflict between reasoning complexity and edge constraints, we adopt the design strategy illustrated in Fig. 1: *object-centric reasoning derived from compact 3D structure*. Instead of retaining dense geometry or relying on heavyweight multi-view aggregation, we distill the instance-segmented point cloud into a sparse, object-level relational graph. The edges of this graph encode a minimal set of local spatial cues, such as relative placement and proximity, that are sufficient to ground common indoor spatial language.

This abstraction serves as a highly efficient interface for reasoning: we prompt a small, instruction-tuned [2] Large Language Model (LLM) with this structured text to solve referring segmentation and scene-language tasks. Crucially, this approach operates *zero-shot*, requiring neither task-specific LLM fine-tuning nor expensive 2D feature extraction. By strictly bounding the representation to schema-constrained instance IDs, we ensure reliable parsing and controlled decoding costs.

To translate this architectural efficiency into practical deployability on constrained edge GPUs, we incorporate a GPU-centric superpoint pooling/clustering pipeline that eliminates CPU bottlenecks during compact scene construction. Furthermore, we develop an edge-oriented LLM inference stack that exploits prefix/KV caching and leverages low-precision quantized model variants. These optimizations significantly reduce prompt prefill costs and recover memory headroom, resulting in a modular system where the 3D segmenter, graph granularity, and LLM size can be dynamically adjusted to meet application-specific accuracy, latency, and energy budgets.

We evaluate our approach across standard indoor bench-

This work was supported in part by the Center for Cognitive Multispectral Sensors (CogniSense), one of the centers in the Joint University Microelectronics Program (JUMP), a SRC program sponsored by the DARPA

marks for referring segmentation and spatial reasoning, reporting both task accuracy and rigorous end-to-end deployment metrics (latency, peak memory, and energy) on an NVIDIA Jetson edge platform. The results demonstrate that compact object relations serve as an effective substrate for zero-shot LLM reasoning, suggesting more broadly that structuring the interface between perception and language can be as critical as model scale in enabling efficient and compositional reasoning. Our system achieves competitive accuracy, outperforming strong baselines in reasoning segmentation, while delivering significant speedups and energy reductions, demonstrating that complex spatial reasoning is achievable within the strict resource constraints of the network edge.

This paper makes the following contributions:

- **An object-centric, token-bounded scene abstraction for zero-shot spatial reasoning.** We propose an object-level, sparse relational representation of indoor 3D scenes designed for LLM prompting. This formulation minimizes input size and reliance on expensive 2D feature extraction while preserving sufficient local spatial context to generalize zero-shot across diverse tasks, including standard referring segmentation, complex multi-hop reasoning, and out-of-distribution queries.
- **GPU-resident compact scene construction for edge deployment.** We integrate a fully GPU-based superpoint pooling and partitioning pipeline that eliminates host-side preprocessing and CPU–GPU synchronization overhead, enabling low-latency perception-to-representation conversion under edge constraints.
- **An integrated edge inference design with quantified speed and energy gains.** We combine prefix/KV caching and low-precision LLM variants (FP4) into a unified inference stack. On an NVIDIA Jetson, we demonstrate a **3.2× speedup** and **1.5× peak memory reduction**, alongside a **1.6× improvement in energy efficiency**, all while maintaining competitive accuracy without fine-tuning.

Our code, configuration files, and evaluation scripts are available at https://github.com/yathAg/CORE3D.

II. Background and Related Work

Indoor scene–language tasks [3], [4], such as 3D referring segmentation and multi-hop referring expressions, are inherently *compositional*: target objects are specified through relations over spatial structure (*e.g.*, left-of, in-front-of, nearest), rather than solely through category recognition. Many practical deployments further require *interactive*, closed-loop behavior (*e.g.*, assistive robotics and human-in-the-loop automation), where responsiveness and latency directly affect usability and safety. Deploying such capabilities on edge platforms, therefore, emphasizes predictable end-to-end latency, memory footprint, and energy per query under batch-1 workloads. In these settings, the LLM stage is often *prefill-dominated*: scene descriptions and constraints are long while outputs are short, making prompt length and KV-cache allocation first-order drivers of runtime and memory behavior. Moreover, on memory-bandwidth-limited devices, moving weights and activations can dominate both time and energy, further amplifying the cost of verbose prompts and high-precision weight storage.

Prior work spans two broad directions, each exposing a different trade-off. First, 2D-heavy multimodal pipelines enrich geometry with appearance cues via multi-view rendering and image-language alignment, often improving semantic richness and robustness [5], [6]. However, these methods typically introduce additional encoders and preprocessing stages, increasing memory traffic and complicating deployment when perception and reasoning must *co-reside* within a fixed edge memory budget. Second, 3D-to-text or 3D-to-graph interfaces avoid explicit 2D feature extraction by converting geometry into language-compatible inputs [7]. While more deployment-friendly in principle, naive serializations are frequently either token-heavy (inflating prefill and cache footprint) or too lossy to support fine-grained relational grounding, and dense relation sets can cause prompt growth that is difficult to control as scene complexity increases. Separately, optimizations such as quantization, caching [8], and GPU acceleration are often evaluated in isolation; in integrated 3D→representation→LLM systems, their interactions are mediated by shared GPU memory headroom (weights *vs.* KV-cache *vs.* 3D activations) and CPU↔GPU synchronization overhead. As a result, improving any single stage in isolation can shift the bottleneck elsewhere. In contrast, our approach addresses this coupling at the representation level, explicitly co-designing the scene abstraction and reasoning interface to jointly control both computational cost and relational expressivity.

We therefore adopt a systems viewpoint and summarize the practical conditions for edge feasibility: (i) *bounded input growth* with scene size to stabilize prefill cost, (ii) *bounded decode* via a compact, machine-parseable output schema, (iii) minimal CPU↔GPU synchronization in perception-to-representation, and (iv) explicit budgeting of GPU memory across perception tensors, model weights, and KV cache to sustain concurrency and avoid out-of-memory failures. These constraints are exacerbated in indoor workloads, where a scene is repeatedly queried. Accordingly, the following section introduces an object-centric scene interface and a GPU- and cache-aware runtime that, together, operationalize these requirements on edge hardware.

III. Design Methodology

Our pipeline explicitly decouples *perception* from *reasoning*, guided by the observation that object-level relational structure provides a sufficiently expressive yet more efficient substrate for many indoor scene–language tasks than dense geometric or multimodal representations. Given an indoor point cloud, we run a 3D instance segmenter that outputs M object instances, each represented by a point membership set $\mathcal{I}_m$, a semantic label y_m, and a confidence score s_m. All downstream components, compact scene abstraction, and LLM-based reasoning, consume only $\{(\mathcal{I}_m, y_m, s_m)\}_{m=1}^{M}$, enabling plug-and-play substitution of the upstream segmenter and isolating perception optimizations from the reasoning stack. This decoupling is deliberate for edge deployment: end-to-end latency is often governed by per-scene fixed costs (preprocessing, CPU-GPU synchronization, and memory movement) rather than GPU FLOPs alone, so enforcing a compact interface stabilizes later-stage cost and makes performance budgeting tractable.

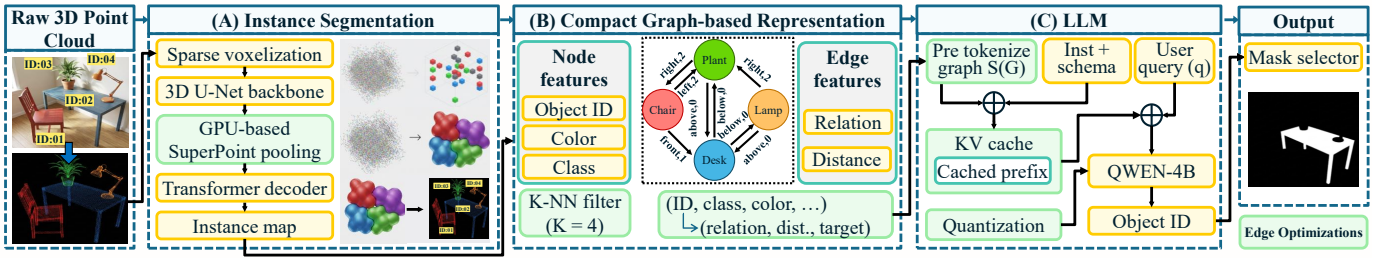

Fig. 2. End-to-end architecture for on-device 3D scene understanding and language-based querying: perception converts raw 3D input into a compact structured scene representation, and a lightweight language model answers user queries from this representation to produce the final output.

The perception stack is trained once and then frozen for all downstream experiments, and the LLM is used with frozen weights throughout.

A. *Efficient Instance Segmentation Backbone*

We adopt an efficient sparse-voxel 3D instance segmentation backbone [9] that follows a voxel-pooling-query pipeline. The input point cloud is voxelized and encoded by a sparse 3D U-Net backbone to obtain point-wise features. To avoid prohibitively expensive per-point transformer decoding, features are flexibly pooled into a reduced set of region tokens (superpoints or voxels), which serve as keys/values for a fixed-size query decoder. The decoder performs self-attention over K queries and cross-attends to the M region tokens, outputting per-query mask kernels and classification scores; region-level masks are produced by convolving region features with the kernels, projecting them back to points via the pooling map, then applying standard thresholding and NMS-style filtering. Overall, pooling-based tokenization controls token growth and substantially reduces memory and decoding cost relative to per-point decoding, making query-based 3D instance segmentation feasible under tight edge budgets.

The resulting perception pipeline (Fig. 2(A)) thus reduces the raw point cloud to a bounded set of instance hypotheses with no host-side preprocessing.

Edge motivation: superpoints as a pooling primitive. Within this design, superpoints play a dual role: they (i) reduce the token count presented to the decoder (reducing latency and KV/memory pressure), and (ii) provide a reusable pooling map that encourages locally coherent aggregation of geometric/semantic features. The practical caveat for edge deployment is that *how* superpoints are obtained matters: standard extraction pipelines are typically CPU-resident preprocessing (often via geometric over-segmentation), which introduces host-side latency, CPU↔GPU synchronization, and a per-scene fixed cost that is hard to amortize in interactive, batch-1 regimes.

GPU-resident superpoint partitioning. To eliminate CPU-bound partitioning overhead, we replace the CPU superpoint extractor with a fully GPU-parallel partitioner adapted from prior work [10], thereby breaking the CPU bottleneck of graph-based superpoint solvers. The module outputs a superpoint assignment for each element and the induced pooling map. This pooling map is used to form region tokens for the downstream query decoder, while leaving the sparse U-Net and decoder architecture unchanged.

Local adjacency graph and learned transition-aware embeddings. The partitioner constructs a sparse adjacency graph over elements and applies a lightweight network to predict a low-dimensional embedding of each component. Edge affinities are derived from embedding similarity and trained with a contrastive objective that encourages high affinity for adjacent same-label pairs (intra-edges) and low affinity for adjacent different-label pairs (inter-edges). These labels correspond to semantic classes and additionally define intra- and inter-edges using instance IDs to better preserve instance boundaries.

Contour-regularized partitioning via GPU-parallel greedy merging. Given the learned affinities, superpoints are obtained by optimizing a contour-regularized partition objective that favors piecewise-constant regions while discouraging discontinuities across strongly connected edges. Instead of sequential CPU merging, the optimization is implemented with GPU-parallel greedy merging steps that propose and apply non-conflicting merges across the adjacency graph until no candidate merges remain. The resulting superpoint assignment directly induces the pooling map for region tokenization, and the query decoder continues to operate on pooled region tokens without architectural changes. This yields a compact, object-centric representation with bounded token count, which we then lift into an explicit instance-level graph for downstream spatial reasoning.

B. *Compact Spatial Representation as a Graph*

We represent each indoor scene as a directed attributed graph $G = (V, E)$, where each node $v \in V$ corresponds to an object instance and each directed edge $(u \to v) \in E$ encodes a local spatial relation. The design goal is a bounded-length, object-centric abstraction that preserves identity and coarse layout while discarding raw geometry that is expensive to store, transmit, or prompt, thereby ensuring that representation size and reasoning cost scale predictably with scene complexity (Fig. 2(B)). To control graph density, we connect each node to its k nearest neighbors in centroid space, yielding $|E| \approx k|V|$ rather than the quadratic growth of a fully connected relation set.

Each node stores only lightweight metadata: an object ID, a semantic class label, a size proxy represented by the number of points assigned to the instance, and a coarse color token computed from the instance's average RGB. The color token is quantized to a small, fixed palette using the Berlin-Kay theory [11], intentionally limiting the vocabulary. This is not aimed at photometric fidelity; instead, it provides a compact appearance cue (*e.g.*, dark *vs.* light chair) that is often sufficient for language queries while avoiding verbose numeric descriptors. This token is an intentionally lossy aggregate and does not require retaining or transmitting raw RGB data.

We first apply the per-scan axis-alignment transform so that all instance centroids are expressed in a consistent, upright world frame, then build a directed k-NN graph over these centroids. For each edge, the direction token is assigned by projecting the centroid offset Δ_{uv} onto a chosen plane (default XZ) and taking the dominant component; optionally, Δ_{uv} is rotated into an observer frame (yaw–pitch–roll, ZYX) to define front/back before labeling.

For each directed edge $(u \rightarrow v)$, we store (i) the metric distance $d_{uv} = \|\mu_v - \mu_u\|_2$ between instance centroids $\mu_u, \mu_v \in \mathbb{R}^3$, and (ii) a discrete direction token in $\{\texttt{left}, \texttt{right}, \texttt{front}, \texttt{back}, \texttt{above}, \texttt{below}\}$ derived from the dominant axis of the relative displacement $\Delta_{uv} = \mu_v - \mu_u$.

We keep both direction and color deliberately low-granularity for two reasons: (1) fast construction, coarse quantization, and dominant-axis labeling are cheap and GPU-friendly; and (2) LLM-friendly semantics, the LLM can resolve many ambiguities from linguistic context and local neighborhood structure, so the representation prioritizes stable, legible cues over precise but verbose geometry.

Finally, we serialize G into a compact, structured text form and use this serialization as the uniform scene interface across tasks. The bounded attribute set and k-NN sparsification jointly constrain prompt growth to $O(|V| + k|V|)$, making the prefill cost predictable on edge devices while retaining a sufficient relational structure for compositional queries, such as "the chair left of the table in front of the sofa."

C. LLM-based Inference and Edge Optimizations

Reasoning interface and canonical scene prefix. For each scene, we serialize the object-relation graph into a text prefix $S(G)$ and append a natural-language query suffix q (Fig. 2(C)). A low-parameter LLM consumes the concatenation $[S(G) \,\|\, q]$ and returns a structured prediction. We define two reusable, global prompt components. *Instruction* denotes a fixed task template that (i) specifies the semantics of the serialized fields (node attributes, relation tokens, distance units, and directed-edge interpretation) and (ii) imposes behavioral constraints (e.g., resolve the query by selecting the most consistent instance(s) from the provided graph). *Schema* denotes a strict, machine-parseable output grammar that constrains responses to either `ID=<int>` for single-object referring or `IDS=[<int>,...]` for multi-object selection, with no additional text. Accordingly, we factor the prompt into the following reusable components:

$$p(G, q) = \underbrace{\text{instruction template} + \text{output schema}}_{\text{global}} \,\|\, \underbrace{S(G)}_{\text{per-scene}} \,\|\, \underbrace{q}_{\text{per-query}}$$

This factorization is motivated by the edge regime, where latency is typically dominated by prompt *prefill* over long scene context. At the same time, the desired output is only a few schema tokens. To keep decoding bounded and evaluation reliable, we enforce the output schema by using conservative, low-entropy decoding with a small `max_new_tokens` and explicit stop strings.

To make the prefix token-stable for caching across repeated queries within a static scene, we serialize $S(G)$ in a deterministic canonical form with a fixed node order and a fixed per-node neighbor order, yielding a token-stable prefix. The serialization is also grounding-oriented: rather than presenting a flat inventory, each node is immediately followed by an indented list of its outgoing relations, colocating spatial cues with the referenced instance and improving parse reliability for relational queries.

Prefix/KV caching for static scenes. Because $S(G)$ is invariant within a scene and token-stable under canonicalization, we compute the attention key/value (KV) states for the prefix once and reuse them across subsequent queries. Let T_{pref} and T_{qry} denote the token lengths of the cached prefix and the query suffix, respectively. Without caching, each query incurs prefill proportional to $T_{\text{pref}} + T_{\text{qry}}$; with prefix/KV caching, subsequent queries prefill only T_{qry}:

$$\text{Prefill}_{\text{no-cache}} \propto (T_{\text{pref}} + T_{\text{qry}}), \qquad \text{Prefill}_{\text{cached}} \propto T_{\text{qry}}$$

In interactive regimes where multiple queries are issued per scene, this amortizes the cost of the dominant prefix computation, reducing end-to-end latency and energy per answered query while highlighting the importance of token-stable, structured scene representations for enabling effective cache reuse.

Low-precision weights for co-resident 3D+LLM inference, cache stability, and energy efficiency. On memory-constrained edge GPUs, LLM weight storage competes with upstream 3D perception allocations (*e.g.*, sparse feature tensors and intermediate activations) and with KV-cache capacity for long scene prefixes. When supported by the runtime, we therefore deploy the reasoning LLM in low-precision weight formats (*e.g.*, FP8, FP4) to reduce the static model footprint and recover memory headroom. This headroom is directly converted into (i) larger and more persistent prefix/KV caches that remain resident across interactive queries, improving reuse and reducing KV-cache eviction and reallocation overheads; and (ii) support for longer graph prefixes and higher-fidelity scenes without out-of-memory failures or forced reductions in scene/graph richness. Beyond capacity, smaller weight representations also reduce memory-bandwidth pressure during inference: fewer bytes moved per layer typically lowers data-movement energy and can improve end-to-end efficiency, which is critical under energy-per-query constraints. Quantization further improves deployability across heterogeneous edge SKUs, including dedicated NPUs and low-power accelerators [12], by enabling the same pipeline to run on lower-memory devices with similar memory-bandwidth and power constraints, rather than requiring a single high-end configuration. Because our tasks use short, schema-constrained outputs, the accuracy impact of low-precision deployment can be cleanly evaluated using ID correctness and downstream segmentation metrics. At the same time, efficiency gains are reflected in peak memory and energy/latency measurements.

Zero-shot, cross-task interface. We use the single instruction + schema contract across datasets and task variants without fine-tuning; task specialization is expressed only through the natural-language query suffix q. This design isolates generalization to language-level composition over the same discrete spatial primitives and prompt structure.

IV. Experimental Setup

A. Datasets and Evaluation Metrics

We evaluate our approach under three regimes designed to stress specific capabilities progressively: (1) **In-distribution grounding** using ScanRefer [13], which establishes a baseline for single-object identification; (2) **Compositional reasoning** using Reason3D [14], which targets complex, multi-hop spatial logic; and (3) **Out-of-distribution (OOD) generalization** using Surprise3D [3]. The latter leverages higher-fidelity ScanNet++ [15] scenes to assess robustness against geometric and relational shifts across six diverse query categories (Commonsense, Human Intention, Narrative Perspective, Parametric Perspective, Relative Position, Absolute Distance).

Across all benchmarks, we report validation results on official splits. To strictly measure zero-shot transfer capability, we retain the exact instruction template and graph serialization without dataset-specific tuning. Performance is quantified using *Acc@0.25* and *Acc@0.50* (hereafter denoted as *A25* and *A50*), where correctness is determined by a 3D box IoU threshold. Additionally, we report mask *mIoU* computed from point-level IoUs, aggregated according to each dataset's official protocol. This unified interface ensures that reported metrics reflect reasoning capacity rather than prompt engineering variances.

B. Implementation Details

To isolate the impact of our reasoning architecture, all experiments utilize a fixed pipeline described in Sec. III with a single, frozen instance segmentation checkpoint, preventing upstream perception variances from confounding the results. Scene graphs are constructed with $k{=}4$ directed nearest neighbors; we employ canonical serialization to ensure token-stable prefixes, which is critical for accurate caching measurements.

We use Qwen3-4B-Instruct [16] served via vLLM [17], configured with a maximum sequence length of *8192* tokens and a generation limit of *256* tokens. To preserve headroom for the 3D perception stage and stabilize peak memory across varying scene sizes, we cap GPU memory utilization at *0.2* to limit KV-cache concurrency to 1 (batch-1, single active sequence).

We evaluate three deployment configurations: a *BF16* reference, *FP8*, and *NVFP4* [18]. For the quantized models, we target Linear layers for both weight and input activation quantization, excluding the LM head. All remaining operations (*e.g.*, norms, embeddings, softmax, residuals) are retained in *BF16*. For each precision setting, we report accuracy alongside end-to-end latency and peak memory to characterize the coupled quality-efficiency trade-off.

C. Edge performance metrics and methodology

We quantify deployability under batch-1 execution on an NVIDIA Jetson Thor platform by reporting (i) end-to-end latency, (ii) a stage-wise latency breakdown (instance segmentation, graph construction, LLM prefill, LLM decode), (iii) peak GPU memory (model weights, runtime allocations, and KV-cache footprint), and (iv) energy per query (J/query) during inference. End-to-end latency is measured over complete query executions using NVIDIA Nsight Systems (`nsys`) profiling traces [19]. For fine-grained attribution within the LLM runtime (e.g., prefill vs. decode), we additionally use vLLM's built-in profiling based on `torch.profiler` [20]. Power telemetry is sampled with `tegrastats`, which reports per-rail instantaneous power on Jetson platforms [21]. Energy per query is computed by time-aligning telemetry with the profiled query interval and integrating power over the aligned interval.

V. Results

A. Primary Quantitative Results

On ScanRefer (Table I), our method is competitive in overall grounding and achieves the best strict-threshold performance among the compared systems, reaching *44.5 A50* and *43.2 mIoU* overall. This improves over the strongest prior baseline by *+2.6 A50* and *+1.2 mIoU*, while maintaining strong performance on the *Unique* subset (*85.9 A50 / 76.0 mIoU*). Performance remains most challenging under the *Multiple* subset (dominant in ScanRefer), where ambiguity and fine-grained relational disambiguation are amplified.

TABLE I
ScanRefer referring segmentation (A25/A50/mIoU)

Method	Unique (∼19%)			Multiple (∼81%)			Overall		
	A25	A50	mIoU	A25	A50	mIoU	A25	A50	mIoU
ScanRefer* [13]	67.6	44.4	39.9	31.2	20.9	19.5	38.2	25.5	23.5
3DVG* [22]	79.5	58.0	49.9	42.0	30.8	27.0	49.3	36.1	31.4
3D-SPS* [23]	84.8	65.6	54.7	41.7	30.8	26.7	50.1	37.6	32.1
3D-LLM* [24]	57.8	30.6	32.5	24.7	12.8	14.0	31.1	16.3	17.6
TGNN [25]	69.3	57.8	50.7	31.2	26.6	23.6	38.6	32.7	28.8
X-RefSeg3D [26]	-	-	-	-	-	-	40.3	33.8	29.9
3D-STMN [27]	**89.3**	84.0	74.5	46.2	29.2	31.1	54.6	39.8	39.5
Reason3D [14]	88.4	84.2	74.6	**50.5**	31.7	**34.1**	**57.9**	41.9	42.0
This work	88.7	**85.9**	**75.51**	38.4	**32.0**	32.2	47.7	**44.5**	**42.56**

Methods marked with * output 3D boxes; masks are reproduced by extracting points inside boxes using the official code.

For reasoning segmentation (Table II), our approach substantially outperforms prior methods, achieving *59.42 A25 / 48.05 A50 / 45.09 mIoU*. Relative to Reason3D, this corresponds to *+16.21 A25*, *+15.95 A50*, and *+13.89 mIoU*, indicating that the object-centric relational graph, coupled with schema-constrained LLM decoding, is particularly effective for multi-hop, relation-heavy queries where purely geometric heuristics and weaker scene abstractions fail.

TABLE II
Reason3D reasoning segmentation (A25/A50/mIoU)

Method	A25	A50	mIoU
OpenScene [5]	24.68	7.14	15.03
OpenMask3D [6]	20.78	6.82	13.38
3D-STMN [27]	25.43	17.78	18.23
Llama2 +CLIP [28], [29]	39.26	25.93	27.23
Reason3D [14]	43.21	32.10	31.20
This work	**59.42**	**48.05**	**45.09**

Finally, on Surprise3D (Table III), our decoupled architecture and minimal spatial cue set generalize strongly, achieving *22.95/13.21 (A25/A50)* and *14.82 mIoU* overall. Relative to the strongest baseline, we improve by *+9.32 A25* and *+5.93 mIoU*; the most significant gains across spatial families indicate

TABLE III
SURPRISE3D PER-CATEGORY AND OVERALL RESULTS (A25/A50/MIOU)

Method	*Knowledge Reasoning*						*Spatial Reasoning*												*Overall Reasoning*		
	Common-sense			Human Intention			Narrative Perspective			Parametric Perspective			Relative Position			Absolute Distance			Overall		
	A25	A50	mIoU	A25	A50	mIoU	A25	A50	mIoU	A25	A50	mIoU	A25	A50	mIoU	A25	A50	mIoU	A25	A50	mIoU
ChatScene [30]	7.86	4.01	-	1.64	1.00	-	0.00	0.00	-	0.00	0.00	-	1.38	0.00	-	1.39	0.00	-	3.59	1.77	-
Intent3D [31]	10.01	3.24	-	15.84	5.82	-	2.65	0.77	-	2.62	0.79	-	4.30	0.97	-	2.41	0.74	-	8.63	2.94	-
Reason3D [14]	6.97	3.11	4.79	11.33	6.03	7.51	8.39	4.88	5.63	7.91	4.82	5.68	9.57	6.38	6.78	9.10	2.60	5.25	9.09	4.57	6.08
3D-Vista [7]	6.14	6.14	-	8.26	8.26	-	6.50	6.50	-	3.65	3.65	-	6.52	6.52	-	7.61	7.61	-	6.40	6.40	-
MLLMfor3D [32]	20.42	**13.42**	12.75	22.38	**13.62**	11.91	15.07	13.62	11.56	4.25	3.20	2.93	7.78	5.81	4.92	11.90	**10.62**	**9.24**	13.63	10.05	8.89
Average	10.28	5.98	8.77	11.89	6.95	9.71	6.52	5.15	8.60	3.69	3.12	4.31	5.91	3.94	5.85	6.48	4.32	7.25	8.27	5.15	7.49
This work	**24.15**	9.52	**14.34**	**25.45**	13.29	**15.49**	**19.92**	**15.27**	**14.40**	**24.86**	**18.80**	**17.66**	**15.75**	**12.60**	**11.82**	**12.73**	9.91	8.73	**22.95**	**13.21**	**14.82**

that sparse, structured cues are sufficient for robust OOD reasoning. Remaining gaps appear primarily at strict thresholds for *Commonsense* (A50) and in *Absolute Distance*, driven by occasional loss of small objects and coarse distance encoding.

B. Latency and Pipeline Efficiency

We evaluate deployability on an NVIDIA Jetson under batch-1 execution, targeting interactive edge budgets. As shown in Fig. 3, the uncached baseline is hindered by CPU-resident superpoint processing and heavy LLM prefill. This results in substantial fixed perception costs that dominate mean latency and severely limit interactive responsiveness.

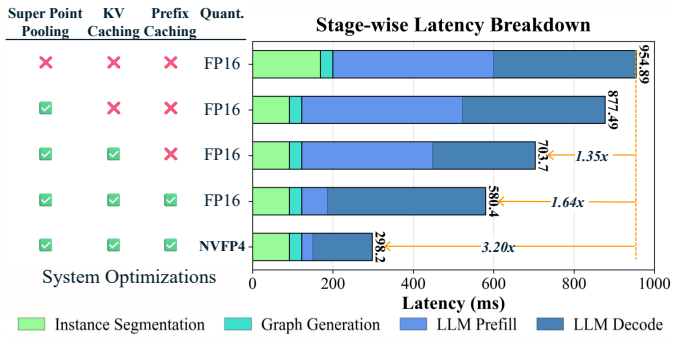

Fig. 3. Latency/energy breakdown (ScanRefer on Jetson) under incremental optimizations (GPU superpoints, KV/prefix reuse, and quantization).

By migrating superpoint partitioning and pooling entirely to the GPU, we eliminate host-side bottlenecks and CPU-GPU synchronization overheads, yielding an immediate *2.3*× reduction in the instance segmentation stage. However, the most significant system-level gains arise from exploiting steady-state scene reuse; specifically, KV cache reuse collapses repeated prefill work, improving mean latency by *1.35*× *vs.* baseline, while prefix-aware reuse further confines inference to a short, schema-constrained decode, resulting in a latency improvement of *1.64*×. Finally, low-precision deployment reduces memory-bandwidth pressure during generation, delivering a *3.20*× end-to-end speedup and placing execution in a predictable regime suitable for real-time indoor querying on constrained edge hardware.

C. Quantization and Memory Impact

Our efficiency analysis demonstrates that quantization is most effective when coupled with prefix/KV reuse. The transition from *BF16* to *FP8* captures the bulk of these gains, yielding a substantial *1.9*× improvement in latency and a *1.6*× reduction in energy consumption. Remarkably, these gains come with essentially no loss in quality (Fig. 4(a)), showing less than *0.5%* degradation, because our decoupled design allows the LLM to reason over compact, token-bounded scene representations rather than raw point clouds. This architecture ensures that upstream perception remains invariant to LLM weight precision, providing a robust foundation for aggressive quantization.

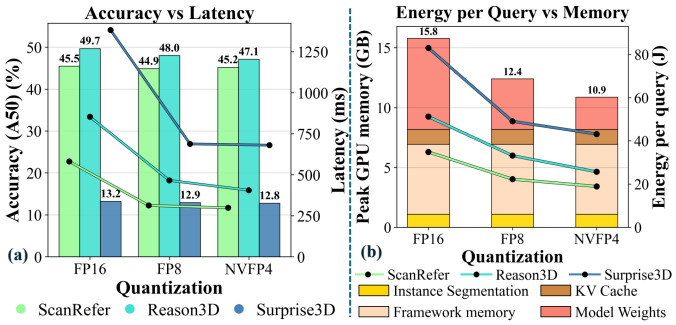

Fig. 4. Quantization trade-offs across datasets. (a) Accuracy (bars) vs. latency (lines). (b) Peak GPU memory (bars) and energy/query (lines).

Further scaling down from *FP8* to *NVFP4* provides more targeted benefits, an incremental *1.07*× reduction in latency and a *1.2*× increase in energy efficiency. However, the primary advantage of this shift is deployment feasibility, as it delivers a vital *1.5*× reduction in peak GPU memory usage (Fig. 4(b)). Since perception and runtime allocations remain constant, reducing the model weights directly reclaims hardware capacity. This memory headroom is the critical factor in preventing out-of-memory failures within complex scenes and supporting the persistent KV cache residency necessary for sustained, low-latency responsiveness in real-time edge intelligence.

D. Qualitative Analysis

We include a compact qualitative analysis to validate that the proposed object-graph prompting captures the compositional cues required by indoor spatial language, while keeping the remaining errors interpretable. Fig. 5(a) highlights the system's capacity for spatial reasoning: notably, the model successfully resolves queries where the target class is not directly present (e.g., deducing a "bed" solely from the functional description "used for sleeping"). This indicates a true understanding of the scene structure and semantics beyond simple label matching. Furthermore, Fig. 5(b) demonstrates that minimal attributes, specifically coarse color and discrete spatial relations, are

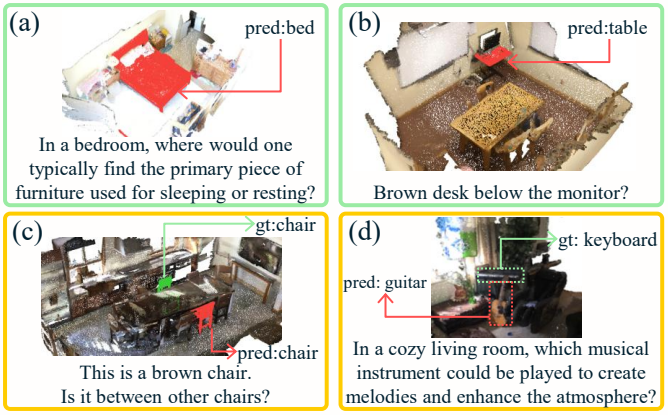

Fig. 5. Qualitative grounding examples showing successes and representative failure modes under the compact object-graph prompt.

sufficient to ground queries like "Brown desk below the monitor?" even when dense geometric features are discarded.

However, this abstraction introduces specific trade-offs. First, the discrete spatial cues can occasionally be too coarse for dense environments. As seen in Fig. 5(c), when multiple same-class instances (e.g., chairs) are clustered with similar local relations, the model may exhibit object-set confusion, failing to distinguish the specific target from its neighbors. Fig. 5(d) highlights cases where failures may stem from the question type or dataset ambiguity rather than a model limitation. For queries relying on fine-grained visual affordances (e.g., "played to create melodies") or commonsense priors, the model may predict a semantically valid but incorrect object (e.g., predicting *guitar* instead of the ground-truth *keyboard*). This suggests that such errors often reflect semantic underdetermination, in which the minimal graph lacks the specific visual texture needed to resolve the ambiguity inherent in the question.

VI. Ablation Studies

A. Effect of Graph Connectivity

Varying the graph density through the number of nearest neighbors (k) reveals a critical accuracy-latency frontier, where adding even minimal relational edges significantly improves grounding over the k=0 baseline. As illustrated in Fig. 6(a), performance gains scale with moderate connectivity, yet the most favorable operating point across all datasets is k=4, where ScanRefer and Reason3D achieve peak accuracy. While Surprise3D exhibits a flatter accuracy trend, increasing connectivity to k=5 densifies the prompt and inflates latency without yielding significant gains, often resulting in negligible or even negative accuracy shifts. Consequently, k=4 serves as our default configuration to maximize reasoning capacity. This sparse relational structure effectively balances the need for compositional spatial cues with the strict token-budget requirements of edge-ready inference.

B. Backbone Sensitivity and Superpoint Portability

To quantify the extent to which end-to-end accuracy is bottlenecked by upstream perception, we conduct a *perfect segmentation* ablation, replacing predicted instance masks with ground-truth labels while keeping the reasoning pipeline fixed.

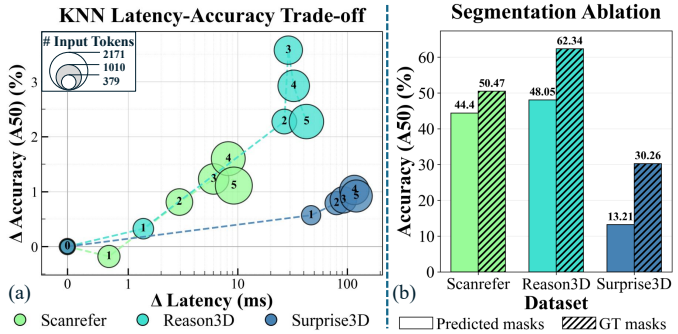

Fig. 6. Ablations. (a) Accuracy/latency/memory vs. graph connectivity (k). (b) A50 with predicted vs. ground-truth instances.

As illustrated in Fig. 6(b), this upper-bound setting consistently improves *A50* across all datasets, revealing significant headroom that our object-graph interface is ready to capitalize on. The dramatic performance jump on Surprise3D is particularly validating; it indicates that the reasoning module successfully resolves complex queries involving rare or long-tail object classes, which standard segmentation models fail to detect. These results indicate that performance is primarily limited by the recall of the upstream segmenter rather than the reasoning architecture itself, suggesting that improvements in perception can be directly translated into downstream reasoning gains within our framework.

TABLE IV
BACKBONE IMPACT ON SCANREFER (A50, AP50, LATENCY). LATENCY IS REPORTED AS TOTAL AND BROKEN DOWN INTO BACKBONE (B), DECODER (D), AND POST-PROCESSING (P).

Backbone	A50	AP50	Peak Mem (MB)	Latency (ms)			
				Total	B	D	P
Minkowski [33]	44.2	39.42	784	100.4	60.0	9.9	30.1
PTV3 [34]	**44.7**	34.78	959	94.8	53.2	8.5	28.9
SpConv [35]	44.5	34.99	**531**	**63.5**	19.1	9.5	30.1

We further validate modularity by swapping the instance segmentation backbone (PTV3, SpConv, Minkowski) while keeping the downstream pipeline fixed [9]. Table IV shows that this decoupling enables efficiency tuning: SpConv achieves the lowest latency and peak memory, reducing total runtime by $\sim 37\%$ relative to Minkowski without degrading downstream reasoning accuracy (A50). While raw instance segmentation quality (AP50) varies across backbones, the stable A50 indicates that our object-centric graph abstracts away minor segmentation artifacts, allowing the use of edge-friendly perception models without sacrificing robust grounding.

C. Text Generation Tasks: Finetuning for Answer Format

We additionally report results on *ScanQA* [36], which requires *free-form text* answers rather than spatial masks and is evaluated with n-gram overlap metrics (CIDEr/BLEU-4). In this regime, the primary zero-shot failure mode is often *answer-format mismatch*: our 3D graph prompt frequently produces semantically correct but verbose or paraphrased responses that do not match ScanQA's canonical short phrases, and are therefore under-credited by automatic metrics.

To disentangle missing evidence from formatting, we fine-tune the language head on ScanQA for 3 epochs using LoRA

($r = 16, \alpha = 32$) across all projection layers. We employ an effective batch size of 32 and a 2×10^{-4} learning rate with a cosine decay schedule, while maintaining the exact 3D-only representation (no RGB/2D features). This process teaches the model concise, dataset-aligned templates, yielding a substantial gain in CIDEr/BLEU-4 and outperforming a strong finetuned baseline that uses 2D cues (Table V). These results support the view that the zero-shot gap on ScanQA is dominated by *style compliance* rather than representational capacity.

TABLE V
SCANQA TEXT GENERATION (CIDER/BLEU-4)

Method	CIDEr ↑	BLEU-4 ↑
Baseline (finetuned + 2D) [30]	87.7	14.3
Ours (zero-shot, no 2D)	33.1	5.5
Ours (finetuned, no 2D)	**93.2**	**15.1**

VII. CONCLUSION

We demonstrate that practical indoor grounding under edge constraints is achievable by decoupling perception from reasoning and using a compact, instance-centric 3D object-relation graph as the interface. Serializing a token-bounded set of object attributes and sparse spatial relations into a stable prefix enables a small LLM to answer referring and relational queries with structured, machine-parseable outputs, avoiding the compute and memory costs of end-to-end multimodal models. Across ScanNet-style grounding and reasoning workloads, the resulting system remains accuracy-competitive while achieving substantial system-level speedups, primarily from GPU-resident superpoint pooling in the perception stack and prefix/KV reuse for repeated queries. These results point to a broader design direction for multimodal systems, where carefully structured interfaces between perception and reasoning can enable both efficiency and generalization, particularly in resource-constrained settings.

REFERENCES

[1] J. Zhang, J. Huang, S. Jin, and S. Lu, "Vision-Language Models for Vision Tasks: A Survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 8, pp. 5625–5644, August 2024.

[2] S. Zhang *et al.*, "Instruction Tuning for Large Language Models: A Survey," *ACM Comput. Surv.*, vol. 58, no. 7, pp. 169:1–169:36, Jan. 2026.

[3] J. Huang *et al.*, "Surprise3d: A dataset for spatial understanding and reasoning in complex 3d scenes," in *The Thirty-ninth Annual Conference on Neural Inf. Process. Syst. Datasets and Benchmarks Track*, 2025.

[4] H. Ding, S. Tang, S. He, C. Liu, Z. Wu, and Y.-G. Jiang, "Multimodal referring segmentation: A survey," 2025, arXiv:2508.00265.

[5] S. Peng *et al.*, "Openscene: 3d scene understanding with open vocabularies," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2023, pp. 815–824.

[6] A. Takmaz, E. Fedele, R. Sumner, M. Pollefeys, F. Tombari, and F. Engelmann, "Openmask3d: Open-vocabulary 3d instance segmentation," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36. Curran Associates, Inc., 2023, pp. 68 367–68 390.

[7] Z. Zhu, X. Ma, Y. Chen, Z. Deng, S. Huang, and Q. Li, "3d-vista: Pre-trained transformer for 3d vision and text alignment," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, October 2023, pp. 2911–2921.

[8] A. Raha *et al.*, "Llm-npu: Towards efficient foundation model inference on low-power neural processing units," in *Proc. IEEE Int. Conf. Omni-layer Intell. Syst. (COINS)*, 2025, pp. 1–8.

[9] M. Kolodiazhnyi, A. Vorontsova, A. Konushin, and D. Rukhovich, "Oneformer3d: One transformer for unified point cloud segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, June 2024, pp. 20 943–20 953.

[10] L. Geist, L. Landrieu, and D. Robert, "Ez-sp: Fast and lightweight superpoint-based 3d segmentation," 2025, arXiv:2512.00385.

[11] B. Berlin and P. Kay, *Basic Color Terms: Their Universality and Evolution*. Berkeley, CA: University of California Press, 1969.

[12] A. Das, Y. Agarwal, S. K. Ghosh, A. Raha, and V. Raghunathan, "Demo Abstract: ECO: Low Power Context-Aware Multimodal AI on NPUs," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, August 2025, pp. 1–2.

[13] D. Z. Chen, A. X. Chang, and M. Nießner, "ScanRefer: 3D Object Localization in RGB-D Scans Using Natural Language," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham: Springer International Publishing, 2020, pp. 202–221.

[14] K.-C. Huang, X. Li, L. Qi, S. Yan, and M.-H. Yang, "Reason3d: Searching and reasoning 3d segmentation via large language model," in *Proc. Int. Conf. 3D Vision (3DV)*, 2025, pp. 1177–1186.

[15] C. Yeshwanth, Y.-C. Liu, M. Nießner, and A. Dai, "Scannet++: A high-fidelity dataset of 3d indoor scenes," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2023, pp. 12–22.

[16] A. Yang *et al.*, "Qwen3 technical report," 2025, arXiv:2505.09388.

[17] W. Kwon *et al.*, "Efficient memory management for large language model serving with pagedattention," in *Proceedings of the 29th symposium on operating systems principles*, 2023, pp. 611–626.

[18] E. Alvarez *et al.*, "Introducing NVFP4 for efficient and accurate low-precision inference," NVIDIA Technical Blog, 2025, jun. 24, [Online]. Available: https://developer.nvidia.com/blog/introducing-nvfp4-for-efficient-and-accurate-low-precision-inference/.

[19] NVIDIA, *NVIDIA Nsight Systems User Guide*, accessed: Jan. 31, 2026.

[20] vLLM Contributors, *Profiling vLLM*, accessed: Jan. 31, 2026.

[21] NVIDIA, *Tegrastats Utility — NVIDIA Jetson Linux Developer Guide*, 2025, jetson Linux r38.2.1. Accessed: Jan. 31, 2026.

[22] L. Zhao, D. Cai, L. Sheng, and D. Xu, "3dvg-transformer: Relation modeling for visual grounding on point clouds," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2021, pp. 2928–2937.

[23] J. Luo *et al.*, "3d-sps: Single-stage 3d visual grounding via referred point progressive selection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2022, pp. 16 454–16 463.

[24] Y. Hong *et al.*, "3D-LLM: Injecting the 3D World into Large Language Models," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36. Curran Associates, Inc., 2023, pp. 20 482–20 494.

[25] P.-H. Huang, H.-H. Lee, H.-T. Chen, and T.-L. Liu, "Text-Guided Graph Neural Networks for Referring 3D Instance Segmentation," *Proc. AAAI Conf. Artif. Intell. (AAAI)*, vol. 35, no. 2, pp. 1610–1618, May 2021.

[26] Z. Qian, Y. Ma, J. Ji, and X. Sun, "X-RefSeg3D: Enhancing Referring 3D Instance Segmentation via Structured Cross-Modal Graph Neural Networks," *Proc. AAAI Conf. Artif. Intell. (AAAI)*, vol. 38, no. 5, pp. 4551–4559, Mar. 2024.

[27] C. Wu *et al.*, "3D-STMN: Dependency-Driven Superpoint-Text Matching Network for End-to-End 3D Referring Expression Segmentation," *Proc. AAAI Conf. Artif. Intell. (AAAI)*, vol. 38, no. 6, pp. 5940–5948, Mar. 2024.

[28] H. Touvron *et al.*, "Llama 2: Open foundation and fine-tuned chat models," 2023, arXiv:2307.09288.

[29] A. Radford *et al.*, "Learning transferable visual models from natural language supervision," in *Proc. of the 38th International Conference on Machine Learning*, ser. Proc. of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 8748–8763.

[30] H. Huang *et al.*, "Chat-scene: Bridging 3d scene and large language models with object identifiers," *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 37, pp. 113 991–114 017, 2024.

[31] W. Kang, M. Qu, J. Kini, Y. Wei, M. Shah, and Y. Yan, "Intent3d: 3d object detection in RGB-d scans based on human intention," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2025.

[32] J. Huang *et al.*, "Mllm-for3d: Adapting multimodal large language model for 3D reasoning segmentation," 2025, arXiv:2503.18135.

[33] C. Choy, J. Gwak, and S. Savarese, "4d spatio-temporal convnets: Minkowski convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 3075–3084.

[34] X. Wu *et al.*, "Point transformer v3: Simpler faster stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2024, pp. 4840–4851.

[35] Spconv Contributors, "Spconv: Spatially sparse convolution library," GitHub, 2022, accessed: Jan. 29, 2026.

[36] D. Azuma, T. Miyanishi, S. Kurita, and M. Kawanabe, "Scanqa: 3d question answering for spatial scene understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, June 2022, pp. 19 129–19 139.