SemanticXR: Low Power, Real-time Queryable Semantic Mapping with a Device-Cloud Architecture

Rahul Singh, Devdeep Ray, Connor Smith and Sarita Adve

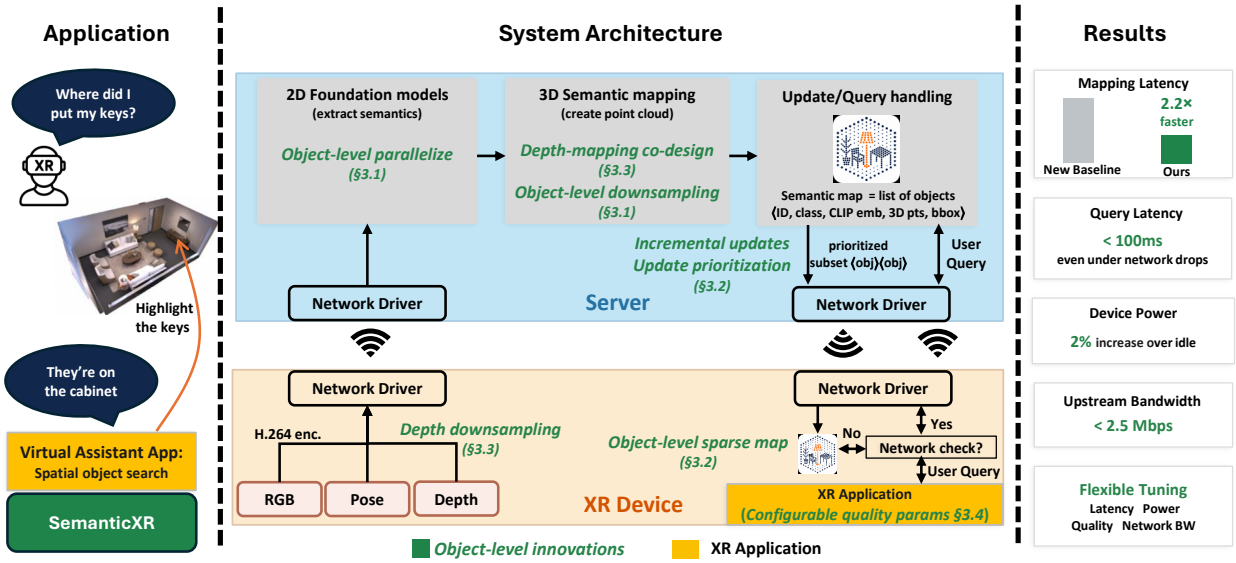


Figure 1: SemanticXR overview. SemanticXR enables real-time, open-vocabulary semantic mapping for low-power XR through a device-cloud architecture organized around objects as first-class units of communication, execution, and memory footprint. Object-level innovations (green) speed up server-side mapping (Sec. 3.1) and reduce upstream bandwidth (Sec. 3.3); on the device, they add a sparse local map with incremental updates and update prioritization for network-robust querying (Sec. 3.2). Organizing every operation at object granularity allows both applications and the system to trade off resource usage against semantic quality, suiting application requirements and operating conditions respectively, without modifying the perception and mapping pipeline (Sec. 3.4).

Abstract—Semantic mapping is a core service that enables grounded interactions in emerging Extended Reality (XR) applications such as AI assistants. Deploying this capability on mobile XR devices requires a system that is open-vocabulary, real-time, and low-power. Existing approaches are compute-intensive and assume server-class resources. Cloud offloading offers a practical path, but no existing system splits semantic mapping between the device and the cloud, and current approaches do not address how to manage communication, execution, and memory footprint across the device-cloud boundary. We present SemanticXR, the first device-cloud system for real-time, open-vocabulary semantic mapping and querying under XR power, bandwidth, and memory constraints. Our key insight is to elevate semantically identifiable objects to first-class units of system design, governing how the system communicates, executes, and manages memory across the device and the server. Evaluation against a new, aggressive device-cloud baseline shows that object-level system organization improves server-side mapping latency by 2.2× at equivalent semantic quality. Object-level depth-mapping co-design maintains upstream bandwidth under 2.5 Mbps. On the device, an object-level sparse local map with incremental updates and update prioritization enables sub-100 ms query latency for up to 10,000 objects even under network drops, supports tens of thousands of objects within 500 MB memory footprint, and scales downstream bandwidth with map changes rather than total scene size. The system adds only about 2% to idle device power.

Index Terms—Extended reality, semantic mapping, open-vocabulary perception, device-cloud systems, low-power real-time systems.

1 Introduction

Extended Reality (XR) has the potential to transform areas such as education, healthcare, accessibility, and industrial work. To support emerging capabilities such as spatial object search, AI assistants, and context-aware scene understanding, XR devices must go beyond reconstructing 3D geometry to associating semantic meaning with the physical environment. This requires a semantic mapping service that incrementally builds and retains a queryable 3D map linking geometry with meaning. For example, when a user asks "Where are my keys?," the system should be able to guide them to the keys' location, even if the keys are not currently in view, and highlight the keys once they come into view.

Deploying semantic mapping as a service on XR devices imposes several concurrent requirements. From the algorithm side, because XR devices operate in diverse and previously unseen environments, the semantic layer must be open-vocabulary, supporting recognition beyond fixed categories. From the system side, the system must operate in real time and within strict power budgets to support interactive applications on battery-constrained, all-day wearable devices. The resulting system

• *Rahul Singh and Sarita Adve are with the University of Illinois Urbana-Champaign. E-mail: rahuls10@illinois.edu, sadve@illinois.edu.*
• *Devdeep Ray and Connor Smith are with NVIDIA. E-mail: devdeepr@nvidia.com, cosmith@nvidia.com.*

Manuscript received xx xxx. 202x; accepted xx xxx. 202x. Date of Publication xx xxx. 202x; date of current version xx xxx. 202x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.202x.xxxxxxx

must deliver high semantic quality, but quality demands vary across applications, creating opportunities to trade off resource usage for semantic coverage and geometric detail.

Recent algorithmic work in vision and robotics has advanced open-vocabulary semantic mapping by lifting outputs from 2D foundation models into persistent 3D representations [5,12,18,22,31,32,45,53,58]. Some of these approaches achieve real-time performance, but they assume server-class GPUs, with compute and power resources well beyond those available on mobile XR devices. No existing system delivers open-vocabulary, real-time semantic mapping within the power constraints of mobile XR devices.

One on-device alternative is specialized hardware acceleration, but rapidly evolving foundation model architectures limit the effective deployment lifetime of such accelerators. Cloud offloading offers an alternate practical path: it leverages powerful server-side GPUs potentially without increasing device power, avoids dependence on custom hardware, and frees on-device resources for other latency-sensitive XR tasks [6,8–10,19,20,24,25,28,38,49,61,62]. However, how to partition semantic mapping across the device and the cloud under XR constraints remains an open problem.

Tab. 1 lays out the design space and system requirements. A natural device-cloud split is to perform both mapping and querying on the server, but this leaves the device unable to answer queries during network drops, which are common in mobile XR. An alternative is to perform mapping on the server and maintain a copy of the semantic map on the device for local querying. This restores query availability during network drops, but introduces new costs: updating the device map requires transferring the full scene to the device, causing downstream bandwidth and device memory footprint to grow with scene size. On the server side, existing approaches incur high per-frame mapping latency despite access to powerful GPUs (Sec. 5.1). No single architecture satisfies all system requirements for deploying semantic mapping under XR constraints.

The underlying limitation is how existing approaches organize computation. Many fuse semantics into monolithic scene-level representations such as global volumetric maps [58], tying every cost to the total scene size and making them fundamentally incompatible with device-cloud deployment. Others detect and operate on fine-grained entities such as objects [12, 18, 32, 52, 53], but target algorithmic quality rather than system organization. In both cases, existing work specifies how to construct semantic maps, but not how to manage their communication, execution, and memory footprint across a device-cloud boundary.

We present SemanticXR, the first end-to-end device-cloud system that enables real-time, open-vocabulary semantic mapping and querying within the power, bandwidth, and memory constraints of mobile XR.[1] Our key insight is to elevate semantically identifiable objects to first-class units of device-cloud system design, governing communication, execution, and memory footprint across the device and the server. This object-level system organization addresses the gaps identified in Tab. 1, and is not tied to a specific foundation model, generalizing across pipelines that produce per-object mapping representations (Sec. 7.1).

Tab. 2 summarizes SemanticXR's object-level system innovations and the system requirements they address. Object-level parallelism and geometry downsampling manage server-side computation at object granularity, improving real-time mapping latency over frame- or scene-level execution (Sec. 3.1). Object-level depth-mapping co-design downsamples depth before transmitting to the server and mitigates quality loss through per-object mapping decisions, providing a lightweight alternative to compression techniques [41] for reducing upstream bandwidth with negligible device-side overhead (Sec. 3.3). On the device, an object-level sparse local map bounds device memory and downstream bandwidth while enabling queries under network drops (Sec. 3.2). Object-level incremental updates keep downstream bandwidth proportional to map changes rather than total scene size (Sec. 3.2). Object-level update prioritization further reduces device memory usage and downstream bandwidth by sending and storing only relevant object updates to the device. Object-level configurable resource usage vs. quality trade-offs unify these innovations, allowing both applications and the system to adapt semantic mapping behavior to application requirements and operating conditions respectively, without modifying the perception and mapping pipeline (Sec. 3.4).

Together, these innovations lead to the following contributions:

1. We introduce SemanticXR, the first system to enable real-time, open-vocabulary semantic mapping and querying within the power, bandwidth, and memory constraints of mobile XR devices.
2. We identify objects as the core system abstraction for device-cloud semantic mapping, elevating them to first-class units of communication, execution, and memory footprint management. This abstraction enables the innovations (Tab. 2) that collectively address all the gaps identified in Tab. 1.
3. We enable per-object configurable resource usage vs. quality trade-offs, allowing diverse applications and the system to adapt semantic mapping to application requirements and operating conditions respectively, without modifying the perception and mapping pipeline.

Since no existing system implements device-cloud semantic mapping, we construct a device-cloud baseline that uses the same perception models and mapping algorithm as SemanticXR but does not organize system operations at object granularity. This controlled comparison ensures that observed differences are attributable to system design rather than algorithmic or model choice. As discussed, approaches that fuse semantics into monolithic representations remain architecturally incompatible with device-cloud deployment (Sec. 8). Among compatible approaches, our device-cloud baseline is the only one to achieve real-time mapping latency without sacrificing semantic quality: the only other real-time approach has worse quality, and every approach with comparable quality runs offline (taking seconds to minutes per frame) (Sec. 5.1). Over this device-cloud baseline, SemanticXR improves real-time mapping latency by 2.2× at equivalent semantic quality, maintains upstream bandwidth under 2.5 Mbps, and enables sub-100 ms query latencies even under network drops while supporting tens of thousands of objects within 500 MB. Downstream bandwidth scales with map changes rather than total scene size, and the system adds only about 2% to idle device power.

The SemanticXR implementation is available as open source [17,40] and a demonstration video is available as supplementary material.

2 Background

2.1 Geometric Mapping and Semantic Mapping

Geometric mapping reconstructs a 3D representation of the environment, typically as a mesh, point cloud, or volumetric model, using inputs such as device pose, depth, and RGB frames. These maps support core XR functionalities such as collision detection, occlusion handling, and spatial audio. However, geometric mapping alone cannot assign semantic meaning to the scene; for example, it cannot distinguish or track objects or determine their semantic attributes.

Many emerging XR applications require understanding not only *where* surfaces exist, but also *what* they represent and how they relate to one another. Supporting such applications requires semantic mapping, which augments geometric reconstructions with persistent, queryable, and spatially grounded semantic attributes. Unlike per-frame semantic perception, semantic mapping maintains a persistent map with spatio-temporal consistency, recognizing previously observed objects, associating new observations with existing entities, and updating their attributes in place rather than creating a new map entry for each observation. Thus, realizing semantic mapping in XR requires not only accurate semantic inference, but also a system that manages a persistent semantic map under power, bandwidth, and memory constraints.

2.2 Foundation Models and Open-Vocabulary Semantic Mapping

Foundation models have substantially advanced 2D scene understanding by enabling open-vocabulary recognition and generalization to previously unseen categories. Models for grounded object detection, segmentation, captioning, and vision-language embedding extraction [7, 16, 23, 29, 46, 47, 63] provide strong building blocks for

[1] Although we refer to cloud offloading, SemanticXR also applies to other split-compute configurations; e.g., offloading to a puck or edge server.

System Architecture	Location		System Requirements					
	Mapping	Query	Device Power	Real-Time Mapping	Upstream BW	Query under Network Drops	Downstream BW	Device Memory
All on-device	Device	Device	×	×	N/A	N/A	N/A	×
Straightforward	Cloud	Cloud	✓	∼	✓	×	✓	N/A
device-cloud†	Cloud	Device	✓	∼	✓	✓	×	×
Ours: SemanticXR (co-designed)	**Cloud**	**Cloud + Device**	✓	✓	✓	✓	✓	✓

Table 1: System requirements for deploying semantic mapping under XR constraints. No single existing architecture meets all requirements for device-cloud semantic mapping under XR constraints. SemanticXR's co-designed organization, built on objects as the core abstraction for communication, execution, and memory footprint, meets all of them. ∼ = partially meets; N/A = requirement does not apply (e.g., an all-on-device system uses no network or server). †No existing system implements a device-cloud split for semantic mapping.

SemanticXR Object-Level Innovation	Device Power	Real-Time Mapping	Upstream BW	Query under Network Drops	Downstream BW	Device Memory
Object-level parallelism	✓	✓				
Object-level geometry downsampling	✓	✓				
Object-level depth-mapping co-design	✓		✓			
Object-level incremental updates	✓			✓	✓	
Object-level sparse local map	✓			✓	✓	✓
Object-level update prioritization	✓			✓	✓	✓
Object-level configurable resource usage vs. quality	✓	✓	✓	✓	✓	✓

Table 2: SemanticXR's object-level innovations and the requirement(s) each addresses. Each is enabled by treating objects as first-class units of communication, execution, and memory footprint. Together they let SemanticXR meet every requirement in Tab. 1.

semantic mapping in XR. However, these models produce view-centric 2D predictions and do not natively maintain open-vocabulary semantics over persistent 3D scenes. Recent work addresses this by lifting outputs from 2D foundation models into persistent 3D representations, as described in Sec. 2.3.

2.3 Semantic Mapping using 2D Foundation Models

Semantic mapping using 2D foundation models has been explored in several prior works [5, 12, 18, 31, 53, 56, 58, 59]. These approaches differ in how they structure semantic information. Some embed 2D region semantics [64, 65] into a monolithic 3D representation [58], tying all costs to total scene size. Others organize semantics around identifiable objects [12, 18], producing discrete per-object representations. Both families target mapping quality rather than system organization and mobile XR constraints; neither addresses how the semantic map is managed across a device-cloud boundary.

All the above approaches rely on computationally intensive foundation models, requiring server-class GPUs well beyond the power budget of mobile XR devices. As discussed in the introduction, monolithic representations that tie every cost to total scene size are fundamentally incompatible with device-cloud deployment. We therefore build on the object-based family of pipelines, which produce discrete per-object representations. Our contributions are not tied to a specific foundation model but apply to any pipeline that produces per-object representations (Sec. 7.1).

2.3.1 Semantic Mapping Flow

Fig. 2 illustrates a representative semantic mapping pipeline. At a high level, the pipeline consists of two stages. First, per-frame semantic information is extracted using open-vocabulary models, producing per-object predictions. Second, these predictions are lifted into 3D using depth and camera pose and incrementally associated with existing objects in the map based on spatial and semantic similarity. Transient observations are pruned over time to mitigate noise.

2.3.2 Querying the Semantic Map

Once a semantic map is constructed, users issue object-grounding queries: natural-language descriptions that retrieve the best-matching scene objects (Fig. 2). The system matches a semantic embedding of the query text against per-object descriptors (e.g., using CLIP embeddings and cosine similarity) and returns the best-matching objects along with their 3D representations, which the app can then use to highlight them

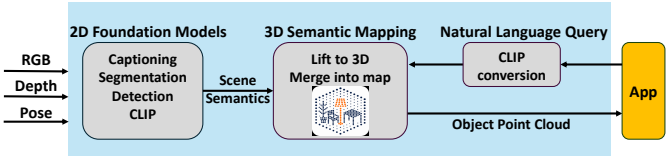


Figure 2: Representative semantic mapping pipeline using 2D foundation models. Per-frame semantic predictions are lifted into 3D using depth and pose, aggregated into a semantic map, and exposed through a query interface.

in the real world. We characterize this query scope and its extension to relational, affordance, and free-space queries in Sec. 7.2. Because queries operate over the persistent semantic map, their cost depends on how that map is maintained and, in device-cloud settings, how much of it must be retained on the device.

3 SemanticXR

SemanticXR builds on the object-based semantic mapping pipelines described in Sec. 2.3 and illustrated in Fig. 2. As discussed, these approaches are largely algorithmic: they target mapping quality on powerful GPUs and are oblivious to the power, bandwidth, and memory constraints of XR devices. As summarized in Tab. 1, no existing approach addresses how to manage communication, execution, and memory footprint across a device-cloud boundary. SemanticXR is a device-cloud system that bridges this gap, providing real-time, open-vocabulary semantic mapping and querying for low-power XR devices.

The key insight of SemanticXR is to elevate semantically identifiable objects to first-class units of device-cloud system design, governing how semantic information is communicated, executed, and stored across the device and the server. A map object is identified by a stable object ID and consists of semantic and geometric descriptors: a semantic embedding, a class label, a 3D point cloud, and a bounding box. This abstraction generalizes across pipelines that produce per-object representations.

As illustrated in Fig. 1, the XR device streams synchronized RGB and depth frames of the user's physical space along with the associated device pose to the server, where the semantic mapping pipeline (Sec. 2.3) detects objects, extracts semantic embeddings, and incrementally associates observations with existing objects and adds

any freshly observed objects to the map. When the network is available, queries (Sec. 2.3.2) are evaluated against the full server-side map. During a network outage, the device falls back to a local semantic map maintained through object-level updates from the server.

This object-level system organization enables the innovations summarized in Tab. 2 and detailed in the following subsections: object-level parallelism and geometry downsampling for improved server-side mapping latency (Sec. 3.1), an object-level sparse local map with incremental updates and update prioritization for network-robust querying with bounded device memory and downstream bandwidth (Sec. 3.2), depth-mapping co-design for lightweight upstream bandwidth reduction (Sec. 3.3), and per-object configurable resource usage vs. quality trade-offs that address communication, execution, and memory footprint (Sec. 3.4).

3.1 Mapping Latency

Open-vocabulary semantic mapping composes multiple foundation models followed by incremental 3D association, resulting in substantial server-side computational load. When execution is organized at frame or scene granularity, all computation is treated uniformly regardless of per-object variation in size and complexity, limiting opportunities for parallelism and missing potential efficiency gains available from this variability.

Object-level parallelism. SemanticXR structures server-side execution at object granularity. After object proposals are generated for a frame, subsequent processing is performed independently for each detected object (Fig. 1). By moving away from a frame-level execution, segmentation and vision-language feature extraction are parallelized across objects within a frame, improving GPU utilization and reducing per-frame processing latency.

Object-level geometry downsampling. Projecting objects into 3D produces highly variable per-object point cloud sizes, depending on the size of the object, leading to disproportionate computational cost. Per-object geometry is needed to place each object in 3D, both for associating and merging observations by spatial proximity and for returning an object's location in response to queries. Because object association and merging depend on spatial proximity and semantic similarity but not on high-fidelity geometric detail, SemanticXR caps the number of points per object through geometry downsampling, bounding per-object computation and improving mapping latency without degrading semantic quality. A system parameter controls this bound, enabling applications to adjust the trade-off between geometric detail and mapping latency (Tab. 3).

Together, object-level parallelism and geometry downsampling improve server-side mapping latency.

3.2 Query Under Network Drops: Downstream Bandwidth and Device Memory Footprint

Interactive semantic queries in XR must remain responsive even during network drops. SemanticXR supports two query modes: *Server Querying (SemanticXR-SQ)*, where queries are evaluated against the full server-side semantic map, and *Local Querying (SemanticXR-LQ)*, where queries are executed on the device using a local semantic map. Enabling SemanticXR-LQ requires maintaining a semantic map on the device, but a full copy of the server map is impractical: device memory would grow with scene size, and keeping the local map updated would require transferring the full map, causing downstream bandwidth to grow with scene size as well. SemanticXR addresses both constraints through three object-level innovations.

Object-level sparse local map. SemanticXR maintains a local semantic map on the device, organized as a collection of per-object entries. Each entry stores the same fields as a server-side object, but with the point cloud further downsampled from the server-side representation (Sec. 3.1) to fit device memory constraints. Downsampling reduces a retrieved object's geometric detail, not which objects a query retrieves. Query accuracy therefore is unaffected and the retained geometry remains sufficient for spatial localization. Because each object's geometry is capped at a configurable point budget rather than stored as a dense scene-wide reconstruction, per-object memory is fixed and total device memory grows only with the number of retained objects, not with scene complexity. An application configurable system parameter controls the point budget per object, enabling applications and the system to adjust the memory–quality trade-off (Tab. 3). The number of retained objects is further bounded by update prioritization, described below.

Object-level incremental updates. Updates to the local map are transmitted as object-level incremental updates. Rather than transferring the full semantic map, the server sends newly created objects together with existing objects whose geometry and semantic descriptors were revised under new viewpoints, each keyed by its stable object ID. As a result, downstream bandwidth is proportional to the number of changed objects rather than the total scene size. Updates are issued periodically and only after an object has been consistently observed across multiple frames, filtering out transient detections before they propagate to the device. A system parameter controls update frequency, enabling applications and the system to balance map freshness against downstream bandwidth (Tab. 3).

Object-level update prioritization. SemanticXR prioritizes which objects are maintained on the device using a configurable set of priority classes, semantic relevance, and proximity to the user (Tab. 3); for example, an application can favor task-relevant categories, nearby objects, or distant landmarks. This allows the system and applications to limit the local map to the most relevant content and balance it against device memory and downstream bandwidth. When the local map reaches its memory budget, admitting a higher-priority update evicts the lowest-priority retained objects, keeping the map within budget. When no priority classes or distance threshold is set, the system instead evicts the objects farthest from the user. The object-level abstraction makes it straightforward to add richer replacement policies in the future.

Query mode switching. During network drops, pending updates are buffered on the server and applied upon reconnection. Network quality is monitored using latency and transmission error signals from the RGB-D stream. When network latency exceeds a configurable threshold (Tab. 3), the system switches from SemanticXR-SQ to SemanticXR-LQ. SemanticXR-LQ may therefore operate on a slightly stale state, but staleness is bounded by the most recent successful update.

3.3 Upstream Bandwidth

Offloading semantic mapping requires the XR device to transmit synchronized RGB and depth frames of the user's physical space, along with the associated device pose, to the server. While RGB can be efficiently compressed using hardware video encoders available on mobile XR devices, depth is typically produced as high-precision frames that cannot leverage the same hardware directly [41]. While prior work has explored depth compression [41], SemanticXR instead asks a co-design question: how much can depth be approximated while preserving semantic quality? This leads to a lightweight alternative – downsampling the depth frames prior to transmission and mitigating quality loss through per-object mapping decisions.

Object-level depth-mapping co-design. While depth downsampling substantially lowers transmission cost, it can degrade geometric detail if applied uniformly. Because the semantic map is organized around discrete objects, mapping decisions are made per object rather than per frame. Objects that occupy sufficient image area retain reliable depth even after downsampling and are incorporated immediately, while smaller or distant objects are deferred until additional observations improve depth reliability. A system parameter controls the minimum object area required for incorporating an observation, enabling applications to adjust the trade-off between upstream bandwidth and semantic quality (Tab. 3).

3.4 Object-Level Configurable Resource Usage vs. Quality

XR applications impose diverse requirements on semantic maps; e.g., some require broad scene coverage while others prioritize specific object categories or regions. Because SemanticXR organizes all system operations at object granularity, applications can trade off resource usage, including compute, memory, and bandwidth, against semantic quality per object or per category, without modifying the underlying mapping pipeline. Tab. 3 summarizes the exposed parameters and their system-level effects. This configurability cuts across all system operations: applications can adjust geometric detail vs. mapping la-

Configurable Trade-off	Exposed System Knobs	Effect and Trade-off	Tuned values
Query latency vs. device power	`net_latency_switch_threshold`	Switches between SemanticXR-SQ and SemanticXR-LQ; trades device power for latency.	100 ms
Mapping fidelity vs server cost	`skip_mapping_set`, `max_object_points_server`	Selects which classes the server maps and caps their per-object point budget.	Empty, 2000 pts for everything
Local map geometric detail vs. memory	`max_object_points_client`	Caps per-object points for all/selected classes in the on-device map.	200 pts for all classes
Object prioritization vs. memory and bandwidth	`priority_class`, `distance_threshold`	Prioritizes on-device objects by class and distance; evicts the lowest priority (farthest by default) at the memory budget.	None, NULL
Local map freshness vs. downstream bandwidth	`local_map_update_frequency`	Controls the frequency of incremental local map updates, trades map freshness for downstream bandwidth.	Every 2 frames
Upstream bandwidth budget	`min_mapping_bbox_area`, `depth_downsampling_ratio`	Defers objects below a projected-area threshold and downsamples depth before transmission.	2000 px and 5×downsampling

Table 3: Application configurable trade-offs and corresponding system-level tuning knobs in SemanticXR. Tuned values correspond to the configuration used in SemanticXR.

tency via geometry downsampling (Sec. 3.1), control local map update frequency vs. downstream bandwidth (Sec. 3.2), and reduce upstream bandwidth via depth-mapping co-design while preserving semantic quality (Sec. 3.3).

4 Evaluation Methodology

4.1 SemanticXR Implementation Details

We evaluate SemanticXR using the object-based semantic mapping pipeline described in Sec. 2.3. Per-object semantic observations are generated using off-the-shelf zero-shot models for captioning (RAM [63]), object detection (Grounding DINO [29]), instance segmentation (MobileSAM [60]), and vision-language embedding (MobileCLIP [54]). The device transmits RGB, depth, and pose to the server (Sec. 3.3), where all semantic mapping is performed. During network drops, the device executes queries against its local semantic map (Sec. 3.2). We use the tuned parameters in Tab. 3 except when studying a specific tunable parameter (e.g., depth downsampling in Sec. 5.5).

4.2 Device-Cloud Baseline and Evaluation Goals

Our evaluation characterizes the effect of object-level system organization on device-cloud deployment under XR constraints. Since no existing system implements device-cloud semantic mapping, we construct a controlled and competitive device-cloud baseline to isolate this effect.

Splitting semantic mapping requires managing communication, execution, and memory footprint across the device-cloud boundary. Monolithic approaches that fuse semantics into global representations [58] tie all three costs to total scene size, incurring prohibitive overhead. Other pipelines [12, 18, 32, 52, 53] produce discrete per-object map representations, which are a prerequisite for the object-level system organization introduced by SemanticXR. Our device-cloud baseline follows this family (Fig. 2) and uses identical perception models and mapping pipeline as SemanticXR (Sec. 4.1) but does not organize system operations at object granularity. The server processes frames without object-level parallelism or geometry downsampling. The device receives periodic full scene updates to support local querying, rather than incremental and sparse object-level transfers. Both SemanticXR and the device-cloud baseline transmit downsampled depth; the effect of depth-mapping co-design on upstream bandwidth is studied independently in Sec. 5.5. Any observed differences are therefore attributable solely to system organization.

As shown in Sec. 5.1, this device-cloud baseline already matches or exceeds prior open-vocabulary mapping approaches in semantic quality while achieving lower mapping latency on the evaluated (Replica) dataset [51].

4.3 Evaluation System Setup

Our evaluation requires fine-grained power instrumentation and controlled low-power configurations to characterize system behavior under XR constraints. Commercial XR headsets do not expose these capabilities, limiting their suitability for evaluating power-sensitive system design. Therefore, for the bulk of our results (Sec. 5), following prior XR systems work [9, 15, 19], we use an NVIDIA Jetson Orin *configured in low-power mode* as a proxy XR device. The platform provides embedded GPU compute, configurable power envelopes, and detailed system instrumentation. The configuration in Tab. 4 emulates a modern mobile XR headset by limiting GPU TPC count, operating frequencies, and power envelope to levels representative of contemporary XR headset power budgets. While it does not capture a headset's thermal dissipation, memory bandwidth, or display and sensor power, this configuration provides a reasonable approximation of headset-class compute and power for our detailed evaluations. Additionally, we also prototype SemanticXR on a Meta Quest 3 [34] headset and an Apple iPad Pro [2] to demonstrate end-to-end working consumer XR systems, as detailed in Sec. 6.

System Parameters	Low Power	MAXN
CPU Frequency (MHz)	1728	2202
GPU Frequency (MHz)	1020	1301
CPU count	8	12
GPU TPC count	3	8
Idle Power	8.69W	10W
Power Cap	**20W**	**60W**

Table 4: Jetson power configurations. Low power mode emulates a modern XR headset. MAXN represents the maximum power setting supported by Jetson Orin. Low power mode caps power at 20W while MAXN can go up to 60W. Idle power in both cases is high.

For the server side of SemanticXR we use a dedicated workstation equipped with an AMD Ryzen Threadripper 7960X CPU and an NVIDIA RTX 6000 Ada GPU. This configuration represents a contemporary cloud-class server and is held fixed across all experiments.

To study the impact of network conditions on system behavior, we evaluate SemanticXR under three network configurations, with the client over the university WiFi network in all cases: (1) a low-latency configuration with an average network round-trip time (RTT) measured as ∼20 ms, (2) a higher-latency wide-area path at an average measured RTT of ∼66 ms, and (3) complete network outage to evaluate local querying (Sec. 3.2). These settings are used consistently across experiments unless otherwise noted.

4.4 Dataset and Evaluation Scope

For our Jetson-based experiments, we use the Replica dataset [51], as adopted by prior semantic mapping works [12, 32]. Replica consists of diverse indoor room and office scenes and provides synchronized RGB, depth, and camera pose sequences with ground-truth semantic labels, enabling controlled and reproducible evaluation of object-level semantic queries. Our evaluation focuses on indoor scenes. Outdoor environments typically require different perception models, but since SemanticXR does not assume specific model architectures, the same system design applies when paired with suitable outdoor models. We leave empirical validation of outdoor deployment to future work.

4.5 Evaluation Metrics

Our evaluation quantifies how object-level system organization influences device-cloud execution efficiency, and characterizes SemanticXR's robustness to network drops.

To isolate the impact of our system design, specifically whether semantic state is managed at scene or object granularity, all experiments hold the semantic mapping pipeline constant (Sec. 4.1).

We evaluate SemanticXR along dimensions that correspond to the system requirements identified in Tab. 1: device power consumption, server-side mapping latency, upstream bandwidth, query latency under network outage, downstream bandwidth, and device memory. We additionally verify that object-level organization preserves semantic quality, so that these efficiency gains do not come at the cost of mapping accuracy. These metrics collectively characterize how SemanticXR's object-level system organization addresses the requirements of device-cloud semantic mapping (Sec. 5).

4.5.1 Latency: Semantic Mapping and Queries

Semantic mapping latency measures the time required by the server to process a single RGB, depth, and pose frame and update the semantic map. This includes semantic perception using foundation models as well as subsequent object-level map fusion. To report throughput (Frames Per Second), we follow standard practice from the literature [43, 44, 53]— semantic maps are incrementally refined and do not require updates for every input frame, so we sample keyframes at a fixed interval and report throughput as total input frames divided by total keyframe processing time. We use a keyframe interval of 5; prior work commonly uses intervals of 10 or higher [44, 53].

Local query latency comprises text embedding extraction and similarity computation against one stored embedding per object (Sec. 2.3.2); it scales with the number of objects and is independent of the per-object geometry stored on the device. Server-side query latency additionally includes the network round-trip to transmit the query and return the retrieved object's geometry.

4.5.2 Semantic Quality

We evaluate semantic quality as object-grounding accuracy: how well a natural-language query retrieves the correct objects from the semantic map. We follow the evaluation methodology of [18] on the Replica dataset [51], which provides ground-truth semantic labels and object annotations. Ground-truth labels are used to generate text queries that are issued against the semantic map constructed by SemanticXR. Retrieved object point clouds are compared against ground-truth using mean class recall (mAcc) and frequency-weighted mean Intersection-over-Union (F-mIoU). These metrics are standard in prior work [12, 18, 32].

4.5.3 Local Map Scalability

To evaluate how the local semantic map scales with scene complexity, we measure device memory and query latency as a function of the number of objects stored on the device. Synthetic semantic maps are constructed by incrementally inserting object point clouds with associated vision-language embeddings. We evaluate map sizes starting from 80 objects (comparable to room-scale Replica scenes) to 1,000, 5,000, and 10,000 objects, representing increasingly large indoor environments. We additionally report results for 25,000 and 50,000 objects to characterize extreme cases.

4.5.4 Network Bandwidth Usage

We measure upstream bandwidth as the average data rate required to stream RGB, depth, and pose from the XR device to the server, accounting for RGB compression and configurable depth downsampling, where the depth downsampling ratio denotes the per-dimension reduction in depth resolution (e.g., $5\times$ per dimension, or $25\times$ overall). Downstream bandwidth measures the volume of semantic map data transferred during map synchronization.

4.5.5 XR Device Power Consumption

We measure power consumption on the XR client device using *tegrastats* [39], sampled at 1 ms intervals. Tegrastats reports power across major subsystems, including CPU, GPU, SoC, DRAM, and system I/O. All reported power measurements include the device's idle power, which is approximately 8.69 W on the Jetson Orin in low-power mode. Except where noted, all SemanticXR experiments run in low-power mode (Tab. 4). We refer to querying at one query every three seconds as *intermittent*, a heavy but plausible interactive rate, and to sustained querying at the maximum achievable rate as *continuous*, an unrealistic worst case. We evaluate device power under the following configurations:

1. **On-device semantic mapping and querying**: Measures total power when the semantic mapping pipeline executes entirely on the device in maximum power mode, an unrealistic deployment that we report only to quantify the power cost that SemanticXR's cloud offloading eliminates.
2. **Server-side mapping with intermittent querying**: Measures device power during normal operation, when semantic mapping runs on the server and the device streams sensor data and issues intermittent queries (SemanticXR-SQ).
3. **Intermittent local querying under network outage**: Measures device power when the device answers intermittent queries (at a heavy rate) against its local map during a network drop (SemanticXR-LQ).
4. **Intermittent local querying at scale**: Measures how device power under intermittent SemanticXR-LQ grows with the number of objects in the local map, using the synthetic local maps from the local-map scalability experiment.
5. **Continuous (unrealistic) local querying**: Although users do not issue queries continuously, some applications may generate short bursts of quick queries. We report continuous querying as a worst-case upper bound on device peak power during a network outage, not a typical usage pattern.

5 Results

This section evaluates the object-level system innovations introduced by SemanticXR (Tab. 2) along the evaluation dimensions defined in Sec. 4.5. All experiments compare SemanticXR against the device-cloud baseline described in Sec. 4.2; observed differences are attributable to system organization rather than algorithmic or model choice. We first establish that the device-cloud baseline is competitive with prior work in mapping latency and semantic quality. We then evaluate how object-level system organization affects server-side mapping latency (Sec. 5.1), query latency under varying network conditions (Sec. 5.2), device memory and query latency for large local maps (Sec. 5.3), downstream bandwidth (Sec. 5.4), upstream bandwidth and its interaction with semantic quality (Sec. 5.5), and device power consumption (Sec. 5.6).

5.1 Semantic Mapping Latency and Quality

Tab. 5 compares semantic quality and per-frame mapping latency across representative state-of-the-art open-vocabulary mapping approaches, our device-cloud baseline, and SemanticXR. Most of these approaches are offline, requiring seconds to minutes per frame. Our device-cloud baseline attains semantic quality competitive with the strongest of them, OpenMask3D, and exceeds all others, while being the only system that operates in real time. SemanticXR matches this quality while improving on the baseline in latency and on Clio-online, the only other real-time method, in the latency and quality trade-off.

Fig. 3 shows how object-level parallelism and geometry downsam-

Method	mAcc	F-mIoU	Latency (s/frame) ↓
ConceptFusion [18]	24.16	31.31	Offline
ConceptGraphs [12]	38.72	35.82	Offline
OpenMask3D [52]	39.54	49.26	Offline
Clio (batch)† [32]	37.95	36.98	Offline
Clio (online)‡ [32]	N/R (bad)‡	N/R (bad)‡	∼0.30*
Our device-cloud baseline	37.0	47.23	0.57
SemanticXR	37.6	48.29	0.26

Table 5: Open-set 3D semantic segmentation accuracy and per-frame mapping latency on eight Replica [51] scenes. Our device-cloud baseline is defined in Sec. 4.2. †Clio-batch reports quality after offline post-processing, which the authors note is required for adequate quality on Replica. ‡N/R–Not reported. Clio-online does not report quality metrics; the authors note that quality degrades significantly even compared to Clio-batch without post-processing. *Reported on different hardware; we do not explore further since quality is not acceptable.

pling reduce per-frame mapping latency over this already competitive device-cloud baseline. Object-level parallelism enables concurrent execution of segmentation and vision-language feature extraction across objects, while geometry downsampling bounds per-object point cloud size. Together, average per-frame mapping latency drops from approximately 570 ms to approximately 260 ms, a 2.2× improvement. Under the throughput methodology described in Sec. 4.5.1, SemanticXR achieves approximately 20 FPS compared to approximately 8.75 FPS for the device-cloud baseline, at equivalent semantic quality (Tab. 5).

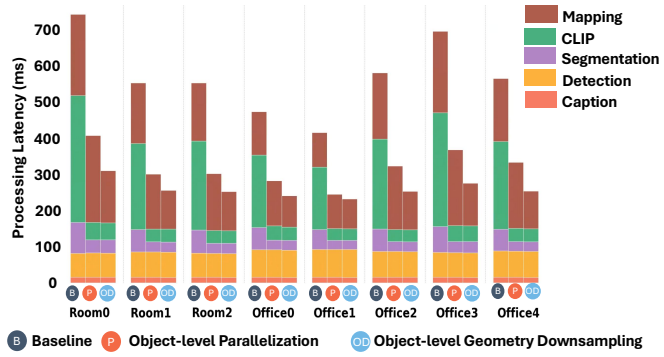


Figure 3: Server-side semantic mapping latency across 8 scenes from the Replica dataset. For each scene, the three bars form an ablation that adds one optimization at a time—device-cloud baseline (B), B + object-level parallelism (P), and B + P + object-level geometry downsampling (OD)—so each optimization's incremental latency reduction is visible. Each bar is decomposed by pipeline stage.

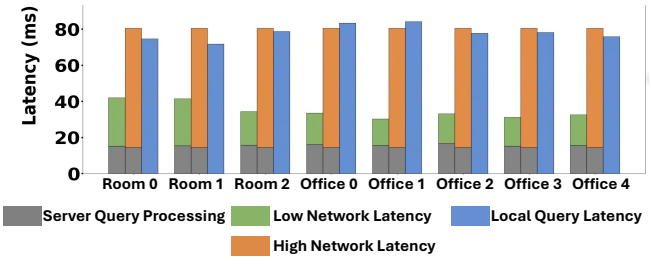


Figure 4: Average query latency for server-side queries (SemanticXR-SQ), decomposed into server processing and network round-trip under the two connected network conditions, and for local queries (SemanticXR-LQ) under outage, across Replica scenes. Minor variations across scenes arise from differences in map size and object density.

5.2 Network Robust Querying

Fig. 4 reports query latency across Replica scenes for server-side (SemanticXR-SQ) and local (SemanticXR-LQ) querying under the three network conditions of Sec. 4.3. When connected, the device uses SemanticXR-SQ, which, under low network latency, is the fastest option and draws the least device power, since text embedding runs on a server-class GPU. As network latency rises, round-trip time dominates and grows variable, and SemanticXR-SQ climbs to meet or exceed SemanticXR-LQ.

SemanticXR-LQ runs entirely on the device, so its latency is flat across all conditions, and it stays available during a network outage, when SemanticXR-SQ cannot run. Because query cost depends only on the number of objects and their embeddings, and not on the geometry stored per object (Sec. 4.5.1), SemanticXR-LQ incurs the same query latency and device power as the device-cloud baseline's full on-device map, while using a far smaller memory footprint (Sec. 5.3).

5.3 Local Map Scalability: Device Memory and Query Latency

We evaluate device memory footprint and local query latency as the number of objects in the local semantic map increases. Fig. 5 shows both metrics for progressively larger synthetic maps ranging from 80 objects (comparable to room-scale Replica scenes) to 50,000 objects. Because SemanticXR's object-level sparse local map caps per-object geometry at a configurable point budget rather than storing a dense scene-wide reconstruction (Sec. 3.2), per-object memory is bounded and total device memory grows with the number of retained objects rather than scene complexity.

SemanticXR supports local maps containing up to 50,000 objects within a 500 MB memory footprint; the associated resource-quality trade-offs are configurable via the parameters in Tab. 3. Fig. 5 also shows the device-cloud baseline's device memory, which retains a dense full-resolution copy of the scene and grows far faster, exceeding 1.1 GB at 50,000 objects. Even under the default configuration (Tab. 3), which prioritizes objects by proximity with no class overrides, SemanticXR's footprint stays within the configured memory budget, so no eviction is triggered; for a scene that reaches the budget, the farthest objects are evicted to keep the map within it (Sec. 3.2).

Local query latency consists of two components: text embedding extraction, which is independent of map size, and similarity computation against stored per-object embeddings, which grows with the number of objects. For maps containing up to 10,000 objects, end-to-end local query latency remains below 100 ms, enabling network-independent querying at interactive latency. Beyond 10,000 objects—a scale reached only in unusually large environments—query latency grows but stays acceptable.

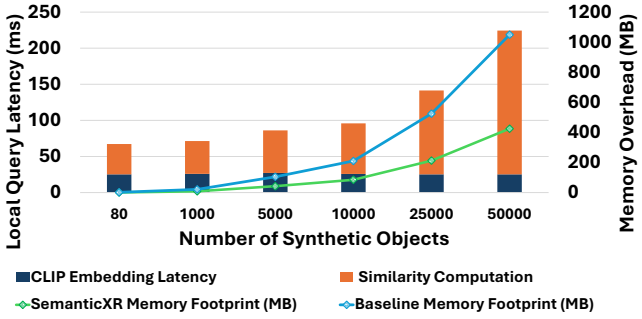


Figure 5: Local query latency and device memory footprint as a function of the number of synthetic objects. Query latency is object-count-bound and identical for SemanticXR and the device-cloud baseline, so a single set of bars is shown. Device memory, however, grows far faster for the device-cloud baseline, which stores the full scene at full geometric resolution, than for SemanticXR's sparse per-object map.

5.4 Downstream Bandwidth

The device-cloud baseline transmits the full semantic map to the device on every update, causing downstream bandwidth to grow with the total number of objects in the scene regardless of how many have changed. In contrast, SemanticXR transmits object-level incremental updates (Sec. 3.2), sending only newly created or modified objects.

Fig. 6 shows this effect. The device-cloud baseline transfers the full scene at full geometric resolution on every update, so its downstream bandwidth grows with the total number of mapped objects. Seman-

ticXR instead transfers only the changed objects, each in the sparse, downsampled form stored in its local map (Sec. 3.2), so the gap reflects both effects: fewer objects per update and a smaller per-object payload. As exploration converges and fewer new objects are discovered, SemanticXR's per-update cost decreases further, whereas the device-cloud baseline's remains at its plateau.

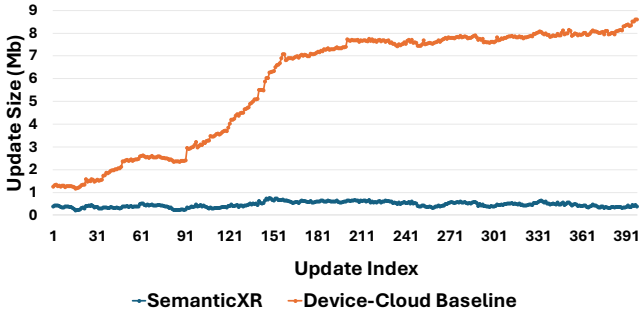


Figure 6: Per-update downstream transfer size as a function of update index for a Replica scene. The device-cloud baseline transfers the full scene at full geometric resolution on each update, so transfer size grows with the total mapped objects until the scene is fully explored (plateau). SemanticXR's object-level incremental updates transfer only changed objects, so per-update cost remains small and decreases as exploration converges. The gap thus reflects both effects: fewer objects per update and a smaller per-object payload.

5.5 Upstream Bandwidth

As discussed in Sec. 3.3, SemanticXR explores a co-design question: how aggressively can depth be downsampled before transmission while preserving semantic quality? Rather than applying compression [41], SemanticXR downsamples the depth frame and mitigates the resulting geometric loss through per-object mapping decisions that defer objects whose small projected area makes their depth unreliable until additional observations are available. Tab. 6 reports the resulting upstream bandwidth and semantic quality when varying depth downsampling ratio and minimum object area (Tab. 3). Reducing depth resolution by 5× in each spatial dimension (25× overall) decreases upstream bandwidth by approximately 90% while causing only a small drop in quality (F-mIoU). Further reductions yield diminishing bandwidth savings while increasingly deferring object integration, as fewer objects meet the minimum area threshold per frame.

Depth Down-sampling	Upstream BW (Mbps)	Quality (F-mIoU)
No Down-sampling	26.4	48.29
2× row, 2× column (4×)	7.72	45.5
3× row, 3× column (9×)	4.26	44.26
4× row, 4× column (16×)	3.06	45.73
5× **row,** 5× **column** (25×)	**2.5**	**45.81**

Table 6: Upstream bandwidth and semantic query quality (F-mIoU) under different depth resolutions. A 5× reduction in each spatial dimension reduces bandwidth by approximately 90% with a small impact on semantic query accuracy and is used as the default configuration. Tab. 5 reports the full-resolution quality (top row).

5.6 XR Device Power Consumption

Fig. 7(a) reports device power across SemanticXR's configurations alongside the on-device alternative it replaces. Running the full pipeline on-device draws 60 W in maximum power mode and takes several seconds to map a single frame, confirming that on-device mapping is impractical within XR power budgets and motivating SemanticXR's cloud offloading. With mapping offloaded, normal operation (server-side mapping with intermittent SemanticXR-SQ) draws 8.87 W, only ∼2% over the 8.69 W idle power (Tab. 4), as the device only streams sensor data and receives results.

Under a network outage the device falls back to intermittent SemanticXR-LQ on its local map, drawing 8.96 W, about 0.3 W (∼3%)

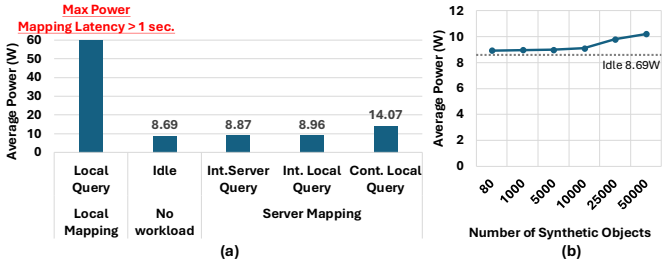
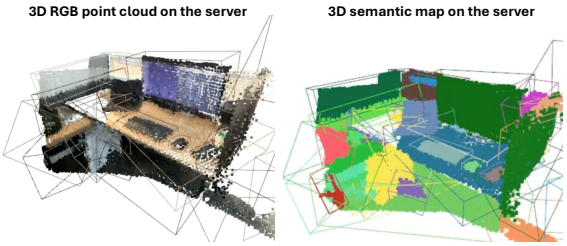


Figure 7: XR device power on Jetson. (a) Average power by workload, grouped by mapping location and query type (Int.: intermittent; Cont.: continuous; Server/Local Query denote SemanticXR-SQ/SemanticXR-LQ). (b) Intermittent local-query power for SemanticXR as a function of the number of synthetic objects, under network outage.

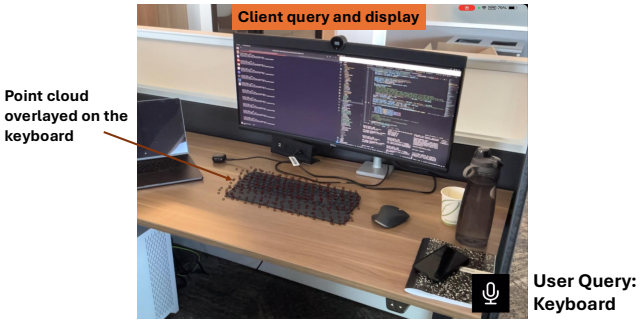
over idle. Device power here is set by CLIP embedding and similarity computation, so it grows with the number of objects rather than how the map is stored: Fig. 7(b) shows it rising from 8.96 W at 80 objects to 10.2 W at 50,000, staying within XR budgets even for very large maps. To characterize worst-case behavior, we additionally measure power under continuous query execution at the maximum achievable rate of 14.7 queries per second, an unrealistic rate for user-facing XR that we report only as an upper bound. Peak power reaches 14.07 W, remaining within the power envelope of contemporary XR devices.

5.7 Summary

Object-level system organization enables real-time open-vocabulary semantic mapping with low-power XR devices. Server-side mapping runs 2.2× faster than a new aggressive baseline at equivalent semantic quality, and queries return in under 100 ms for up to 10,000 objects even when the network drops. SemanticXR makes network-robust cloud offloading feasible by reducing the upstream bandwidth by 90%, decoupling the downstream bandwidth from the scene size, and holding tens of thousands of objects under 500 MB of memory. All of this costs ∼2-3% power on top of idle, whether the device queries the server or its local map.



(a) Semantic mapping: Left: Reconstructed color point cloud on the server. Right: 3D semantic map represented as object-based point cloud.



(b) iPad client view: The user presses the mic button to submit a voice query for a keyboard. The iPad receives the keyboard point cloud and overlays it on the physical keyboard.

Figure 8: SemanticXR deployed end-to-end: (a) Overview of semantic 3D scene mapping. (b) Query output. This figure is created with an iPad as the client. The accompanying video shows a demonstration with a Quest 3 as the client.

6 Deployment Case Study: Quest and iPad Prototypes

To validate SemanticXR with real sensors, we built end-to-end prototypes on two consumer platforms: a Meta Quest 3 [34] mixed-reality headset and an Apple iPad Pro [2]. On the iPad, the client obtains RGB, depth, and pose through ARKit [3]; on the Quest 3, it obtains RGB and depth through Meta's Passthrough Camera and Depth APIs and pose from headset tracking, all in Unity [35,36]. Deploying on an actual XR headset in addition to the iPad shows that SemanticXR's lightweight device-side runtime ports across mobile XR sensing stacks.

Each client streams a subset of captured frames to the server to match mapping throughput and exposes SemanticXR's tunable interface. The iPad transmits H.264 RGB (720×1280, 5 Mbps) with pose and depth (144×256); the Quest 3 transmits native-resolution RGB with pose and depth (320×320). For spatial queries, the client streams audio to the server, which transcribes it, queries the semantic map, and returns matching object-level point clouds that the client overlays in the user's view.

Although neither device exposes fine-grained power telemetry (hence our Jetson-based power characterization, Sec. 4.3), these deployments demonstrate end-to-end operation on real mobile hardware, including an actual XR headset, with live sensors and interactive queries. A demonstration video of the Quest 3 deployment is included as supplementary material. Fig. 8 shows the iPad deployment: the server incrementally builds the semantic map (Fig. 8a) and the client overlays queried objects in the user's view (Fig. 8b).

7 Discussion

7.1 Applicability Across Semantic Mapping Pipelines

SemanticXR's system innovations (Tab. 2) are not tied to a specific perception model; they require only that the backend produce persistent, discrete object-level semantic state that can be incrementally maintained. Backends that already maintain per-object representations (e.g., [12, 18, 32, 53]) can adopt SemanticXR's abstractions directly by promoting objects to independently managed system state. Pipelines that aggregate object inference into monolithic scene representations [56, 58] must first expose persistent object entities, requiring more substantial restructuring.

7.2 Query Scope

SemanticXR is evaluated on object-grounding queries, which retrieve the scene objects that best match a natural-language description. The object-level organization extends to other query classes through targeted additions rather than a redesign: multi-object and relational queries via an external reasoning agent operating over the object map, affordance queries via additional per-object attributes, and free-space or global-layout queries via complementary representations such as occupancy.

7.3 Relationship to Model-Level Optimization

Model-level optimization, such as quantization, pruning, and distillation, is orthogonal to SemanticXR's object-level system organization. Because the foundation models run on the server, such optimization reduces server-side mapping latency and does not reduce device power. Object-level system organization is complementary in the way it reduces computation, communication, and memory cost, affecting latency and device power.

7.4 Future Work

Future directions include server-side scalability to multiple concurrent devices, support for dynamic environments with moving objects and multi-user collaborative mapping, and extending the object-level interface to richer geometric representations such as per-object meshes with non-uniform detail. SemanticXR's configurable knobs (Sec. 3.4) also motivate autotuning and dynamic control that build on existing autotuning frameworks [1, 42, 48], automatically adapting resource-quality parameters to application and network conditions instead of manual configuration. Agentic integration is another direction: humans could issue complex spatial tasks that an agent resolves by issuing spatial queries against SemanticXR's map and acting on the returned objects. Privacy-aware design for device-cloud XR systems remains an important open problem. Quantitative power and thermal characterization on commercial XR headsets, together with evaluation in outdoor environments, would further validate deployment generality.

8 Related Work

Prior work has studied semantic mapping primarily as an algorithmic perception problem, while cloud offloading in XR has largely focused on rendering or isolated perception tasks. In contrast, deploying semantic mapping as a low-power, long-running XR service requires managing communication, execution, and memory footprint across a device-cloud boundary, a problem that existing approaches do not address.

8.1 Semantic Mapping in Robotics and XR

A large body of work has explored open-vocabulary semantic mapping by embedding image- or text-derived semantics into 3D representations. Representative systems include Clio [32], SpatialLM [33], OpenFusion [58], ConceptFusion [18], OpenMask3D [52], ConceptGraph [12], and One Map to Find Them All [5]. These approaches differ in perception models, representations, and update strategies, and collectively demonstrate the feasibility of constructing semantic maps from RGB-D observations. Many support online RGB-D processing [32, 58], while others operate offline over preprocessed point clouds. However, these systems target algorithmic quality under server-class hardware assumptions, whether through task-driven scene representations [32] or monolithic scene-level representations [56, 58], rather than system organization for device-cloud deployment under XR constraints.

Unlike these open-vocabulary systems, a related line of robotics work builds real-time hierarchical 3D scene graphs from closed-vocabulary perception, including Hydra [13, 14] and SceneGraphFusion [55]. Because this perception is lightweight, these systems run on a single machine and do not address the device-cloud split that foundation-model perception forces on mobile XR. Clio [32] extends this scene-graph line toward open-vocabulary, task-driven mapping and is the closest comparison among these systems. As shown in Tab. 5, SemanticXR matches Clio-batch quality without offline post-processing and improves on Clio-online in the latency and quality trade-off. Separately, 3D instance segmentation methods [31, 53, 57] focus on geometric shape decomposition rather than persistent, queryable semantic state and do not target device-cloud deployment. In contrast, SemanticXR treats existing semantic mapping pipelines as backends and elevates objects to first-class units of device-cloud system design, governing communication, execution, and memory footprint across the device-cloud boundary (Tab. 2).

8.2 VLM-based Assistants and Scene Understanding

Recent XR systems have explored natural language interaction using VLM-based assistants such as Google's Project Astra [11] and XaiR [50]. These systems reason over visual observations together with short-term textual or embedding-based memory [11], or attach per-frame features to pre-built geometry without distinguishing individual objects [50]. Neither builds a persistent, queryable 3D semantic map with explicit object identities. SemanticXR addresses an orthogonal problem: how to build and manage such a map across a device-cloud boundary under the power, bandwidth, and memory constraints of mobile XR.

8.3 Cloud Offloading in XR

Several prior works have explored cloud offloading for XR to address the compute and power limitations of mobile devices, including offloading components such as head pose estimation, visual tracking, and rendering [4, 8, 9, 19, 26, 30, 38, 41]. These systems primarily focus on latency-sensitive rendering pipelines or geometric reconstruction, and do not consider the challenges associated with maintaining and querying a persistent semantic map. In addition, offloading AI workloads such as object detection and segmentation has been widely studied [6, 24, 25, 28, 49, 61, 62]. While some of these systems operate at object granularity, they typically target task-specific domains (e.g., object detection over a limited and predefined set of categories) and do not address the requirements of open-vocabulary semantic mapping. Techniques such as keyframe sampling reduce bandwidth or server

compute, but do not address the system-level challenges introduced by offloading semantic mapping in XR, including maintaining persistent semantic state and robustness to network variability.

8.4 Semantic Maps on Neural Representations

Recent work has explored semantic scene representations based on neural fields, such as NeRFs [37] augmented with language grounding [22], and Gaussian-based representations [21] extended for semantic queries [27, 45]. While effective for dense reconstruction and semantic querying, these representations typically require optimization over accumulated observations and are not yet designed for incremental, low-latency updates under the power, bandwidth, and memory constraints of mobile XR systems.

Taken together, prior work provides strong semantic mapping backends and cloud-XR mechanisms, but does not address how to manage communication, execution, and memory footprint for semantic mapping across a device-cloud boundary under the power, bandwidth, and memory constraints of mobile XR. SemanticXR addresses this gap through object-level system organization that is compatible with a broad class of existing semantic mapping pipelines.

9 Conclusion

We presented SemanticXR, the first device-cloud system for real-time, open-vocabulary semantic mapping and querying within the power, bandwidth, and memory constraints of mobile XR. Our key insight is that semantically identifiable objects can serve as first-class units of device-cloud system design, governing how the system communicates, executes, and manages memory across the device and the server. This object-level system organization enables a family of innovations (Tab. 2) that collectively address the requirements of XR device-cloud semantic mapping: object-level parallelism and geometry downsampling improve server-side mapping latency by 2.2× at equivalent semantic quality, object-level depth-mapping co-design reduces upstream bandwidth by ∼90% with a small quality loss, and an object-level sparse local map with incremental updates and update prioritization enables sub-100 ms query latency for up to 10,000 objects and holds 50,000 objects within 500 MB, with downstream bandwidth proportional to map changes rather than total scene size. The system adds only ∼2% device power over the idle power. Because SemanticXR operates on object-level semantic state rather than model-specific internals, its system innovations generalize across any semantic mapping pipeline that produces persistent per-object representations.

Acknowledgments

We thank the anonymous reviewers for their constructive feedback. This work was supported in part by the National Science Foundation (NSF) under grants 2120464, 2217144, and 2454014, and by an internship at NVIDIA.

References

[1] J. Ansel, S. Kamil, K. Veeramachaneni, J. Ragan-Kelley, J. Bosboom, U.-M. O'Reilly et al. Opentuner: An extensible framework for program autotuning. In *2014 23rd International Conference on Parallel Architecture and Compilation Techniques (PACT)*, pp. 303–315, 2014. doi: 10.1145/2628071.2628092 9

[2] Apple Inc. Apple iPad Pro, 2020. Tablet device with LiDAR sensor. 5, 9

[3] Apple Inc. Apple ARKit, 2023. Apple's AR Developer tool. 9

[4] A. Behroozi, Y. Chen, V. Fruchter, L. Subramanian, S. Srikanth, and S. Mahlke. Slimslam: An adaptive runtime for visual-inertial simultaneous localization and mapping. In *Proc. of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ASPLOS '24, pp. 900–915. Association for Computing Machinery, 2024. doi: 10.1145/3620666.3651361 9

[5] F. L. Busch, T. Homberger, J. Ortega-Peimbert, Q. Yang, and O. Andersson. One map to find them all: Real-time open-vocabulary mapping for zero-shot multi-object navigation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 14835–14842, 2025. doi: 10.1109/ICRA55743.2025.11128393 2, 3, 9

[6] K. Chen, T. Li, H.-S. Kim, D. E. Culler, and R. H. Katz. Marvel: Enabling mobile augmented reality with low energy and low latency. In *Proc. of the 16th ACM Conference on Embedded Networked Sensor Systems*, SenSys '18, pp. 292–304. Association for Computing Machinery, 2018. doi: 10.1145/3274783.3274834 2, 9

[7] M. Cherti, R. Beaumont, R. Wightman, M. Wortsman, G. Ilharco, C. Gordon et al. Reproducible scaling laws for contrastive language-image learning. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2818–2829, 2023. 2

[8] A. Dhakal, X. Ran, Y. Wang, J. Chen, and K. K. Ramakrishnan. Slamshare: visual simultaneous localization and mapping for real-time multi-user augmented reality. In *Proc. of the 18th International Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '22, pp. 293–306. Association for Computing Machinery, 2022. doi: 10.1145/3555050.3569142 2, 9

[9] S. Gao, J. Liu, Q. Jiang, F. Sinclair, W. Sentosa, B. Godfrey et al. Xrgo: Design and evaluation of rendering offload for low-power extended reality devices. In *Proc. of the 16th ACM Multimedia Systems Conference*, MMSys '25, pp. 124–135. Association for Computing Machinery, 2025. doi: 10.1145/3712676.3714444 2, 5, 9

[10] Google. Immersive stream for xr overview. https://cloud.google.com/immersive-stream/xr/docs/concept, 2022. Google Cloud; accessed 2025. 2

[11] Google. Project astra. https://deepmind.google/technologies/gemini/project-astra/, 2024. Google's universal AR AI agent. 9

[12] Q. Gu, A. Kuwajerwala, S. Morin, K. M. Jatavallabhula, B. Sen, A. Agarwal et al. Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5021–5028, 2024. doi: 10.1109/ICRA57147.2024.10610243 2, 3, 5, 6, 7, 9

[13] N. Hughes, Y. Chang, and L. Carlone. Hydra: A real-time spatial perception system for 3D scene graph construction and optimization. In *Robotics: Science and Systems (RSS)*, 2022. 9

[14] N. Hughes, Y. Chang, S. Hu, R. Talak, R. Abdulhai, J. Strader et al. Foundations of spatial perception for robotics: Hierarchical representations and real-time systems. *The International Journal of Robotics Research*, 2024. doi: 10.1177/02783649241229725 9

[15] M. Huzaifa, R. Desai, S. Grayson, X. Jiang, Y. Jing, J. Lee et al. Illixr: An open testbed to enable extended reality systems research. *IEEE Micro*, 42(4):97–106, 2022. doi: 10.1109/MM.2022.3161018 5

[16] G. Ilharco, M. Wortsman, R. Wightman, C. Gordon, N. Carlini, R. Taori et al. Openclip, July 2021. doi: 10.5281/zenodo.5143773 2

[17] ILLIXR Consortium. ILLIXR: Illinois Extended Reality. https://illixr.org/. 2

[18] K. Jatavallabhula, A. Kuwajerwala, Q. Gu, M. Omama, T. Chen, S. Li et al. Conceptfusion: Open-set multimodal 3d mapping. *Robotics: Science and Systems (RSS)*, 2023. 2, 3, 5, 6, 7, 9

[19] Q. Jiang, Y. Pang, W. Sentosa, S. Gao, M. Huzaifa, J. Zhang et al. Remotevio: Offloading head tracking in an end-to-end xr system. In *Proc. of the 16th ACM Multimedia Systems Conference*, MMSys '25, pp. 101–112. Association for Computing Machinery, 2025. doi: 10.1145/3712676.3714442 2, 5, 9

[20] T. Jin, M. Dasa, C. Smith, K. Apicharttrisorn, S. Seshan, and A. Rowe. Meshreduce: Scalable and bandwidth efficient 3d scene capture. In *2024 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*, pp. 20–30, 2024. 2

[21] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023. 10

[22] J. Kerr, C. M. Kim, K. Goldberg, A. Kanazawa, and M. Tancik. Lerf: Language embedded radiance fields. In *International Conference on Computer Vision (ICCV)*, 2023. 2, 10

[23] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023. 2

[24] Z. J. Kong, Q. Xu, and Y. C. Hu. Arise: High-capacity ar offloading inference serving via proactive scheduling. In *Proc. of the 22nd Annual International Conference on Mobile Systems, Applications and Services*, MOBISYS '24, pp. 451–464. Association for Computing Machinery, 2024. doi: 10.1145/3643832.3661894 2, 9

[25] Z. J. Kong, Q. Xu, J. Meng, and Y. C. Hu. Accumo: Accuracy-centric multitask offloading in edge-assisted mobile augmented reality. In *Proc. of the 29th Annual International Conference on Mobile Computing and Networking*, ACM MobiCom '23, art. no. 30, 16 pp. Association for Computing Machinery, 2023. doi: 10.1145/3570361.3592531 2, 9

[26] Z. Lai, Y. C. Hu, Y. Cui, L. Sun, and N. Dai. Furion: Engineering high-

quality immersive virtual reality on today's mobile devices. In *Proc. of the 23rd Annual International Conference on Mobile Computing and Networking*, MobiCom '17, pp. 409–421. Association for Computing Machinery, 2017. doi: 10.1145/3117811.3117815 9

[27] M. Li, S. Liu, H. Zhou, G. Zhu, N. Cheng, T. Deng et al. Sgs-slam: Semantic gaussian splatting for neural dense slam. In *European Conference on Computer Vision (ECCV)*, pp. 163–179. Springer-Verlag, 2024. doi: 10.1007/978-3-031-72751-1_10 10

[28] L. Liu, H. Li, and M. Gruteser. Edge assisted real-time object detection for mobile augmented reality. In *The 25th Annual International Conference on Mobile Computing and Networking*, MobiCom '19, art. no. 25, 16 pp. Association for Computing Machinery, 2019. doi: 10.1145/3300061.3300116 2, 9

[29] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023. 2, 5

[30] E. Lu, S. Bharadwaj, M. Dasari, C. Smith, S. Seshan, and A. Rowe. Renderfusion: Balancing local and remote rendering for interactive 3d scenes. In *2023 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 312–321, 2023. doi: 10.1109/ISMAR59233.2023.00046 9

[31] S. Lu, H. Chang, E. P. Jing, A. Boularias, and K. Bekris. Ovir-3d: Open-vocabulary 3d instance retrieval without training on 3d data. In *7th Annual Conference on Robot Learning*, 2023. 2, 3, 9

[32] D. Maggio, Y. Chang, N. Hughes, M. Trang, D. Griffith, C. Dougherty et al. Clio: Real-time task-driven open-set 3d scene graphs. *IEEE Robotics and Automation Letters*, 9(10):8921–8928, 2024. doi: 10.1109/LRA.2024.3451395 2, 5, 6, 7, 9

[33] Y. Mao, J. Zhong, C. Fang, J. Zheng, R. Tang, H. Zhu et al. Spatiallm: Training large language models for structured indoor modeling. arXiv preprint arXiv:2506.07491, 2025. 9

[34] Meta Platforms, Inc. Meta Quest 3, 2023. Mixed reality headset with RGB passthrough and depth sensing. 5, 9

[35] Meta Platforms, Inc. Depth API, 2024. Meta Horizon OS developer tool for environment depth in Unity. 9

[36] Meta Platforms, Inc. Passthrough Camera API, 2025. Meta Horizon OS developer tool for Quest camera access in Unity. 9

[37] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: representing scenes as neural radiance fields for view synthesis. *Commun. ACM*, 65(1):99–106, 2021. doi: 10.1145/3503250 10

[38] NVIDIA. Cloudxr. https://www.nvidia.com/en-us/design-visualization/solutions/cloud-xr/, 2023. Streaming for extended reality. 2, 9

[39] NVIDIA. tegrastats Utility. https://docs.nvidia.com/drive/drive_os_5.1.6.1L/nvvib_docs/index.html#page/DRIVE_OS_Linux_SDK_Development_Guide/Utilities/util_tegrastats.html, 2024. 6

[40] NVIDIA Corporation. XR AI. https://github.com/NVIDIA/xr-ai. 2

[41] Y. Pang, S. Kondguli, S. Wang, and S. Adve. Ada: A distributed, power-aware, real-time scene provider for xr. *IEEE Transactions on Visualization and Computer Graphics*, 31(11):9677–9687, 2025. doi: 10.1109/TVCG.2025.3616835 2, 4, 8, 9

[42] Y. Pei, S. Biswas, D. S. Fussell, and K. Pingali. Slambooster: An application-aware online controller for approximation in dense slam. In *2019 28th International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pp. 296–310, 2019. doi: 10.1109/PACT.2019.00031 9

[43] Z. Peng, T. Shao, L. Yong, J. Zhou, Y. Yang, J. Wang et al. Rtg-slam: Real-time 3d reconstruction at scale using gaussian splatting. In *ACM SIGGRAPH Conference Proc., Denver, CO, United States*, 2024. 6

[44] Z. Peng, K. Zhou, and T. Shao. Gaussian-plus-sdf slam: High-fidelity 3d reconstruction at 150+ fps. *Computational Visual Media*, 2025. 6

[45] M. Qin, W. Li, J. Zhou, H. Wang, and H. Pfister. Langsplat: 3d language gaussian splatting. *arXiv preprint arXiv:2312.16084*, 2023. 2, 10

[46] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal et al. Learning transferable visual models from natural language supervision. arXiv preprint arXiv:2103.00020, 2021. 2

[47] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma et al. Sam 2: Segment anything in images and videos. arXiv preprint arXiv:2408.00714, 2024. 2

[48] H. Sharif, Y. Zhao, M. Kotsifakou, A. Kothari, B. Schreiber, E. Wang et al. Approxtuner: a compiler and runtime system for adaptive approximations. In *Proc. of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '21, pp. 262–277. Association for Computing Machinery, 2021. doi: 10.1145/3437801.3446108 9

[49] S. Shi, J. Cui, Z. Jiang, Z. Yan, G. Xing, J. Niu et al. Vips: real-time perception fusion for infrastructure-assisted autonomous driving. In *Proc. of the 28th Annual International Conference on Mobile Computing And Networking*, MobiCom '22, pp. 133–146. Association for Computing Machinery, 2022. doi: 10.1145/3495243.3560539 2, 9

[50] S. Srinidhi, E. Lu, and A. Rowe. Xair: An xr platform that integrates large language models with the physical world. In *2024 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 759–767, 2024. doi: 10.1109/ISMAR62088.2024.00091 9

[51] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green et al. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. 5, 6, 7

[52] A. Takmaz, E. Fedele, R. W. Sumner, M. Pollefeys, F. Tombari, and F. Engelmann. OpenMask3D: Open-Vocabulary 3D Instance Segmentation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. 2, 5, 7, 9

[53] Y. Tang, J. Zhang, Y. Lan, Y. Guo, D. Dong, C. Zhu et al. Onlineanyseg: Online zero-shot 3d segmentation by visual foundation model guided 2d mask merging. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3676–3685, June 2025. 2, 3, 5, 6, 9

[54] P. K. A. Vasu, H. P. Ansari, F. Faghri, R. Vemulapalli, and O. Tuzel. Mobileclip: Fast image-text models through multi-modal reinforced training. In *CVPR*, 2024. 5

[55] S.-C. Wu, J. Wald, K. Tateno, N. Navab, and F. Tombari. SceneGraphFusion: Incremental 3D Scene Graph Prediction from RGB-D Sequences. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 9

[56] C. Xu, R. Kumaran, N. Stier, K. Yu, and T. Höllerer. Multimodal 3d fusion and in-situ learning for spatially aware ai. In *2024 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2024. doi: 10.1109/ISMAR62088.2024.00063 3, 9

[57] X. Xu, H. Chen, L. Zhao, Z. Wang, J. Zhou, and J. Lu. Embodiedsam: Online segment any 3d thing in real time. *arXiv preprint arXiv:2408.11811*, 2024. 9

[58] K. Yamazaki, T. Hanyu, K. Vo, T. Pham, M. Tran, G. Doretto et al. Open-fusion: Real-time open-vocabulary 3d mapping and queryable scene representation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9411–9417, 2024. doi: 10.1109/ICRA57147.2024.10610193 2, 3, 5, 9

[59] Y. Yang, H. Yang, J. Zhou, P. Chen, H. Zhang, Y. Du et al. 3d-mem: 3d scene memory for embodied exploration and reasoning. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17294–17303, June 2025. 3

[60] C. Zhang, D. Han, Y. Qiao, J. U. Kim, S.-H. Bae, S. Lee et al. Faster segment anything: Towards lightweight sam for mobile applications. *arXiv preprint arXiv:2306.14289*, 2023. 5

[61] Q. Zhang, X. Zhang, R. Zhu, F. Bai, M. Naserian, and Z. M. Mao. Robust real-time multi-vehicle collaboration on asynchronous sensors. In *Proc. of the 29th Annual International Conference on Mobile Computing and Networking*, ACM MobiCom '23, art. no. 57, 15 pp. Association for Computing Machinery, 2023. doi: 10.1145/3570361.3613271 2, 9

[62] W. Zhang, Z. He, L. Liu, Z. Jia, Y. Liu, M. Gruteser et al. Elf: accelerate high-resolution mobile deep vision with content-aware parallel offloading. In *Proc. of the 27th Annual International Conference on Mobile Computing and Networking*, MobiCom '21, pp. 201–214. Association for Computing Machinery, 2021. doi: 10.1145/3447993.3448628 2, 9

[63] Y. Zhang, X. Huang, J. Ma, Z. Li, Z. Luo, Y. Xie et al. Recognize anything: A strong image tagging model. *arXiv preprint arXiv:2306.03514*, 2023. 2, 5

[64] Y. Zhong, J. Yang, P. Zhang, C. Li, N. Codella, L. H. Li et al. Regionclip: Region-based language-image pretraining. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16793–16803, 2022. 3

[65] X. Zou, J. Yang, H. Zhang, F. Li, L. Li, J. Wang et al. Segment everything everywhere all at once. In *Proc. of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, art. no. 868, 14 pp. Curran Associates Inc., 2024. 3