A Survey of Spatial Memory Representations for Efficient Robot Navigation

Ma. Madecheen S. Pangaliman[1,2] Steven S. Sison[1] Erwin P. Quilloy[1] Rowel O. Atienza[1]
[1]University of the Philippines Diliman [2]University of Santo Tomas

Abstract

As vision-based robots navigate larger environments, their spatial memory grows without bound, eventually exhausting computational resources, particularly on embedded platforms (8–16 GB shared memory, <30 W) where adding hardware is not an option. This survey examines the spatial memory efficiency problem across 88 references spanning 52 systems (1989–2025), from occupancy grids to neural implicit representations. We introduce the overhead factor $\alpha = M_{peak}/M_{map}$, the ratio of peak runtime memory (the total RAM or GPU memory consumed during operation) to saved map size (the persistent checkpoint written to disk), exposing the gap between published map sizes and actual deployment cost. Independent profiling on an NVIDIA A100 GPU reveals that α spans two orders of magnitude within neural methods alone, ranging from 2.3 (Point-SLAM) to 215 (NICE-SLAM, whose 47 MB map requires 10 GB at runtime), showing that memory architecture, not paradigm label, determines deployment feasibility. We propose a standardized evaluation protocol comprising memory growth rate, query latency, memory–completeness curves, and throughput degradation, none of which current benchmarks capture. Through a Pareto frontier analysis with explicit benchmark separation, we show that no single paradigm dominates within its evaluation regime: 3DGS methods achieve the best absolute accuracy at 90–254 MB map size on Replica, while scene graphs provide semantic abstraction at predictable cost. We provide the first independently measured α reference values and an α-aware budgeting algorithm enabling practitioners to assess deployment feasibility on target hardware prior to implementation.

1. Introduction

Visual simultaneous localization and mapping (SLAM) has been one of the most actively studied problems in computer vision for over two decades [9, 34], underpinning applications from autonomous driving to augmented reality and domestic robotics. Despite substantial progress, a fundamental limitation emerges at deployment scale: ORB-SLAM3 [5] operates within a few hundred megabytes on a five-minute EuRoC [4] sequence, a standard visual-inertial benchmark comprising stereo-camera and IMU recordings in a machine hall and a Vicon room at ETH Zurich, yet mapping an entire office building ($\sim$3,000 m^3) over a workday strains available memory. For neural SLAM systems, a city-block-scale mission can exhaust GPU memory well before completion.

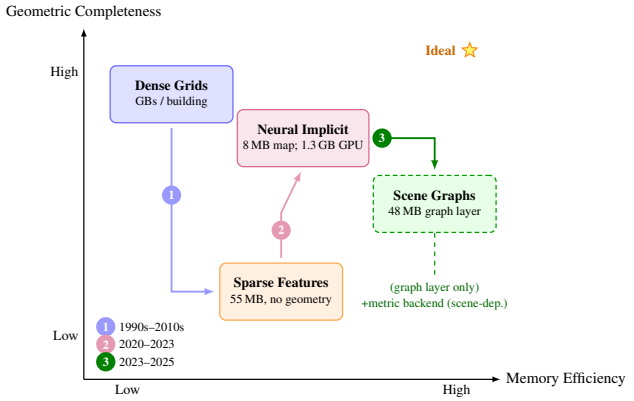

Figure 1. Evolution of spatial memory representations along two competing demands: geometric completeness and memory efficiency. Sparse features (**1**) traded completeness for efficiency; neural methods (**2**) recovered completeness via learned compression; scene graphs (**3**) added semantic abstraction. The scene graph box is dashed to indicate that the 48 MB figure reflects *only* the graph abstraction layer [26]; the required metric-semantic backend [54] adds scene-dependent cost (dashed arrow), shifting the true system footprint leftward. The scene graph's vertical position reflects its semantic completeness (objects, rooms, places) rather than geometric fidelity.

For readers entering this field, the core challenge is straightforward: a robot must simultaneously determine its position (*localization*) and build a map of its surroundings (*mapping*). The *spatial memory* is the persistent data structure encoding this map. Different representations offer distinct tradeoffs between geometric detail, memory cost, and query speed (Fig. 2). This survey organizes these representations along the memory efficiency axis. Readers new to the field can use Fig. 2 as a roadmap, Table 4 to match a

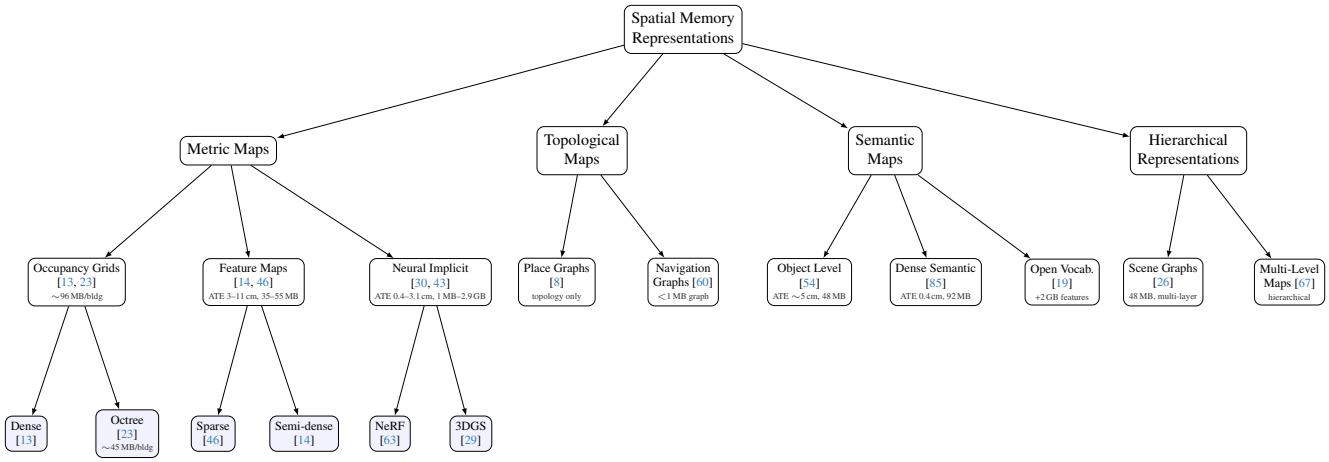

Figure 2. Taxonomy of spatial memory representations with representative citations and typical efficiency metrics (gray). ATE = absolute trajectory error on standard benchmarks; map sizes are representative for building-scale environments. 3DGS = 3D Gaussian Splatting. Semantic maps include open-vocabulary approaches leveraging vision-language models such as CLIP.

representation to hardware constraints, and Algorithm 1 to compute the maximum feasible map size.

This spatial memory growth problem, where representations expand continuously with environment size and mission duration, is the central focus of this survey (Fig. 1). The consequences compound: query latency exceeds real-time deadlines, map updates fall behind the sensor stream, and on resource-constrained platforms, exceeding available RAM terminates the system. Three root causes drive the growth: *spatial extent* (dense representations scale as $O(V)$ with mapped volume), *mission duration* (observations accumulate linearly with time), and *revisitation patterns* (naive approaches store redundant data while more efficient designs exploit repetition for compression).

One might argue that memory is inexpensive and abundant, but deployment platforms present a different picture. Autonomous robots, drones, and AR headsets operate on embedded GPUs with 8–16 GB shared memory (e.g., NVIDIA Jetson Orin) under strict power budgets ($<$30 W), and upgrading hardware on a deployed robot is not feasible. SplaTAM ($\alpha_{\text{GPU}} = 55$) consuming 14 GB at runtime leaves $<$2 GB for perception, planning, and the OS, making the system infeasible despite the map being only 254 MB. Scaling to a 100 m^2 apartment would extrapolate to $\sim$200 GB, far exceeding even a data-center-class A100 (80 GB). Memory efficiency is therefore not a cost problem but a *feasibility* constraint.

Recent surveys [6, 68] organize neural SLAM systems by *method* and benchmark accuracy. Our work is complementary: we organize systems by *memory behavior*, treating neural methods as one instance of a broader memory-scaling problem. This reframing changes practical recommendations: Tosi et al. would rank Co-SLAM favorably based on its 8 MB map, while our α analysis reveals that the same system requires 1.3 GB at runtime ($\alpha_{\text{GPU}} = 157$).

1.1. Contributions

We make the following contributions:

1. We introduce the *overhead factor* $\alpha = M_{\text{peak}}/M_{\text{map}}$, a diagnostic metric that distinguishes α_{CPU} (process Resident Set Size, RSS) from α_{GPU} (device allocation), and compile the first cross-paradigm comparison of runtime versus persistent memory, exposing the gap between published map sizes and actual deployment cost.
2. We independently profile five neural SLAM systems (Co-SLAM, NICE-SLAM, Point-SLAM, SplaTAM, SGS-SLAM) on an NVIDIA A100 GPU, revealing that α_{GPU} varies by two orders of magnitude even within neural methods: from 2.3 (Point-SLAM) to 215 (NICE-SLAM, whose 47 MB map requires 10 GB at runtime). These are, to our knowledge, the first independently measured α_{GPU} values for neural SLAM (Table 3, Fig. 4).
3. We propose a standardized evaluation protocol with four complementary metrics beyond static map-size reporting (Section 5), and an α-aware budgeting algorithm that lets practitioners compute maximum feasible map size from two inputs, namely the memory budget and α, before selecting a representation (Algorithm 1).
4. We present a Pareto frontier analysis with explicit benchmark separation (Fig. 3), a taxonomy of 52 systems across four paradigm families (Fig. 2), memory dynamics categorization, and concrete open challenges with specific research questions.

2. Scope and Methodology

This survey focuses on the *representation layer* of *vision-based* spatial memory: how spatial knowledge from cameras and IMUs is encoded, compressed, and managed over time. Core building blocks (ORB [55], SuperPoint [11],

NeRF [43], 3DGS [30], CLIP) originate in mainstream computer vision, yet their downstream memory implications are rarely discussed. We exclude full autonomy stacks, perception without persistent memory, and LiDAR-based mapping (except BioSLAM [78]). We prioritize breadth; for deeper NeRF/3DGS treatment see [6, 68].

We surveyed papers from 1989 to 2025, filtering for systems that introduce or evaluate persistent spatial memory representations. This yielded 88 references covering approximately 52 distinct systems (15 directly compared in efficiency tables); the remainder are datasets, benchmarks, optimization techniques, and related surveys. Section 3 covers all four representation families (including hierarchical and semantic maps); Section 4 presents our unified efficiency analysis; Section 5 combines the evaluation protocol with practical deployment guidance; Section 6 categorizes memory dynamics; and Section 7 identifies open problems.

3. Spatial Memory Representations: A Memory-Centric Overview

We review the four dominant representation families through the lens of memory efficiency: scaling, α, and compression strategies. Tables 1 and 2 collect comparable metrics; for system-level SLAM contributions see [6, 68].

3.1. Dense Maps: $O(V)$ Scaling

Occupancy grids [13] scale linearly with mapped volume:

$$M_{\text{grid}} = \frac{V}{r^3} \times b \tag{1}$$

where V is mapped volume, r is voxel resolution, and b is bytes per cell. At 5 cm resolution with b=4 B, a 3,000 m^3 building floor requires ∼96 MB, which is already prohibitive for embedded platforms. OctoMap [23] compresses this 2–13× via octree subdivision (∼45 MB at building scale; Table 3); UFOMap [12] and Voxblox [49] (based on Truncated Signed Distance Fields, TSDF; see Table 4) offer further reductions, but the $O(V)$ scaling wall persists for any volumetric representation. No dense-only system reports α; runtime overhead is dominated by sensor processing rather than map maintenance, so α is expected to be low (<10) but remains unverified.

3.2. Sparse Features: Low Overhead, Limited Geometry

Feature-based SLAM [5, 45, 46] trades dense geometry for an order-of-magnitude memory reduction, storing sparse keypoint landmarks, keyframes, and covisibility graphs. Semi-dense methods such as LSD-SLAM [14] reconstruct along image gradients, offering richer geometry at moderate memory increase but without the full $O(V)$ cost of dense grids. A typical building-scale ORB-SLAM3 session yields an estimated ∼55 MB payload with $\alpha_{\text{CPU}} \approx 4$ (180–250 MB peak CPU RSS on EuRoC [4], Intel i7-10700). This low α means map size reliably predicts deployment cost. Visual-inertial systems exhibit similar profiles: ORB-SLAM3 achieves the best ATE (3.5 cm) and the highest composite efficiency (Eff. = 5.2) in Table 1, while Basalt [69] trades accuracy (8.8 cm, Eff. = 3.2) for the smallest map (35 MB) and lowest peak memory (120 MB); VINS-Mono [52] (10.6 cm monocular-inertial, Eff. = 2.4) trails on ATE but operates within 150 MB peak using only a monocular camera. All three sparse systems sustain 20–30 FPS (Table 1), comfortably real-time on a single CPU core.

Table 1. Feature-based visual SLAM comparison on EuRoC [4] (MH01–MH05; sensing modes vary per system; see footnotes). Eff. = $10^3/(\text{ATE} \times M_{\text{map}})$; see Section 4 for caveats on this composite metric.

System	ATE (cm)	Map (MB)	Peak (MB)	FPS	Eff.
VINS-Mono [52]	10.6‖	40*	150*	20*	2.4
Basalt [69]	8.8§	35*	120*	30*	3.2
ORB-SLAM3 [5]	3.5†	55‡	220‡	30	5.2

* Survey estimate from profiling; the cited paper does not report this value directly.
§ Average RMS ATE over MH01–MH05 stereo (proposed VIO); computed from Table I of [69].
† Average RMS ATE across all 11 EuRoC sequences (stereo-inertial) [5]. Note: the 9 mm figure sometimes associated with ORB-SLAM3 refers to TUM-VI, not EuRoC.
‖ Average RMS ATE over MH01–MH05 (monocular-inertial); reproduced by ORB-SLAM3 authors [5], Table II.
‡ Map and peak memory are survey estimates; [5] does not report memory usage.

Two memory management strategies are particularly instructive. RTAB-Map [35] bounds active memory via a working memory / long-term memory transfer: infrequently accessed nodes are moved to disk regardless of session length. MS-SLAM [82] applies sliding-window sparsification, reporting >70% reduction in peak memory increase relative to ORB-SLAM3 while maintaining accuracy. Learned-flow systems (DROID-SLAM [64], 11 GB GPU; DPVO [65]) achieve strong accuracy but exhibit GPU-dominated memory profiles ($\alpha \gg 1$), presaging the overhead patterns observed in neural methods. Table 1 compares systems with comparable memory metrics.

3.3. Neural Maps: High Overhead, Hidden Cost

Neural representations learn compact functions [43]:

$$F_\theta : (\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma)$$

Maps are remarkably small (iMAP [63]: ∼1 MB; Co-SLAM [71]: 8 MB), but runtime GPU memory reveals the true cost: Co-SLAM requires 1.3 GB ($\alpha_{\text{GPU}} = 157$) due to optimizer state, gradient buffers, and rendering caches (analyzed in Section 4). In contrast, iMAP's tiny map yields the highest within-paradigm composite efficiency (Eff. = 320.5, Table 2), despite having the worst ATE

Table 2. Neural SLAM on Replica (synthetic). Eff. = $10^3/(\text{ATE} \times M_{\text{map}})$, higher is better. †: CVPR/ECCV 2024 or later.

System	ATE (cm)	PSNR (dB)	Map (MB)	Peak (MB)	Eff.
iMAP [63]	3.12[f]	22.1[d]	1[a]	—[d]	320.5
NICE-SLAM [86]	1.06[b]	24.4[g]	47[e]	10,082[e]	20.1
Co-SLAM [71]	1.00[h]	30.2[h]	8[e]	1,258[e]	125.0
Point-SLAM [56]	0.52	35.2	2,865[e]	6,563[e]	0.7
SplaTAM† [29]	0.36	34.1	254[e]	14,024[e]	10.9
MonoGS† [42]	0.58	38.9	90[d]	—[d]	19.2
GS-SLAM† [77]	0.50	34.3[h]	198[c]	—[d]	10.1
SGS-SLAM† [39]	0.41	34.7[h]	254[e]	40,330[e]	9.6

[a] Default width-256 network; 1.04 MB per iMAP Table 2.
[b] Avg. ATE RMSE on Replica from Point-SLAM [56] Table 1.
[c] Scene embedding for Replica Room0 from GS-SLAM [77] Table 4.
[d] Survey estimate; not directly reported in the cited paper.
[e] Independently measured on Replica/room0 (NVIDIA A100-SXM4-80GB). Map size from saved checkpoint on disk; peak GPU memory via `nvidia-smi` at 1 Hz, baseline subtracted.
[f] ATE for Room0 only from MonoGS [42] Table 2; avg. across 8 Replica scenes is 2.58 cm. Used here for consistency with single-scene profiling.
[g] Avg. PSNR on Replica from Point-SLAM [56] Table 2 and MonoGS [42] Table 5 (both report 24.42 dB for NICE-SLAM).
[h] From GS-SLAM [77]: ATE from Table 1, PSNR avg. 34.27 dB from Table 6. SGS-SLAM PSNR from SGS-SLAM [39] Table 1.

(3.12 cm) and PSNR (22.1 dB); at the other extreme, Point-SLAM's 2.9 GB map yields Eff. = 0.7, a 458× spread illustrating why we advocate Pareto analysis over scalar ratios. Throughput also varies widely: Co-SLAM reaches 16 FPS, while Point-SLAM runs at only 1 FPS and 3DGS systems range from 2 FPS (SGS-SLAM) to 8 FPS (GS-SLAM; Table 3). Hash-based methods [27, 44, 71] bound M_{map} by construction, but quality degrades silently when scene complexity approaches table capacity, providing bounded memory rather than constant-quality scaling.

3D Gaussian Splatting (3DGS) [30] stores explicit primitives (∼236 B each) scaling as $O(N_G)$ with scene surface area; without management, growth is unbounded. Compression is an active research area: learned masking with codebook quantization [37] achieves 10–25×, pruning with vector quantization [17] ∼15×, and Gaussian merging [2] further reduces count. Parallel work extends 3DGS SLAM along robustness [83], loop closure [84], and semantics [28, 39] axes, but critically none of the recent systems [7, 21, 22, 25, 38, 51, 57, 73] report M_{peak} or α, leaving deployment costs unknown. The memory challenge differs by architecture: compact implicit methods and semantic 3DGS exhibit large gaps between map and runtime cost ($\alpha_{\text{GPU}} = 55$–215 for measured systems; SGS-SLAM's $\alpha = 159$ rivals hash-based Co-SLAM), while dense neural point methods shift the burden to the map itself (Point-SLAM: 2.9 GB, $\alpha = 2.3$; Table 3). Neither regime achieves both compact maps and low runtime overhead.

3.4. Hierarchical and Semantic Representations

Topological maps grow with distinct places rather than volume, offering fundamentally different scaling. FabMap [8] pioneered appearance-based place recognition (topology only, no metric geometry). Dense semantic approaches such as SNI-SLAM [85] jointly reconstruct geometry and semantics (ATE ∼0.5 cm on Replica). Hydra [26] and its successors [81] construct hierarchical scene graphs at 5 Hz on a Jetson Xavier NX (8 GB) with a 48 MB graph abstraction, but this figure excludes the underlying Kimera [54] metric-semantic mesh (ATE ∼5 cm), making the total memory scene-dependent and underreported. Multi-robot extensions [67] further multiply the footprint. The key memory question for scene graphs is *which layers count*: the graph alone is compact, but the metric backend can rival a neural map.

Vision-language features introduce a new memory dimension. CLIP embeddings (d=512, 32-bit) cost ∼2 GB per million points, comparable to the geometric map itself. Three strategies trade off memory against capability: HOV-SG [74] stores features per segment (∼75% reduction), Clio [41] clusters by task relevance, and VLMaps [24] fuses features into a 2D grid. Downstream planning systems (LM-Nav [60], SayNav [53], Open Scene Graphs [40], Embodied-RAG [75]) inherit the map's memory profile and add planning-layer overhead that none quantify, making total system α unknown for semantic navigation.

4. Unified Efficiency Analysis

A key contribution of this survey is a unified efficiency analysis comparing accuracy against memory cost *across* paradigms. We structure this around two complementary metrics.

4.1. The Overhead Factor α

Published map sizes are misleading proxies for deployment cost. As illustrated by Co-SLAM [71] in Section 3.3, a system saving an 8 MB map requires 1.3 GB of GPU memory at runtime. To capture this gap, we define the *overhead factor*:

$$\alpha = \frac{M_{\text{peak}}}{M_{\text{map}}} \tag{2}$$

where M_{peak} is peak runtime memory (CPU RSS or GPU allocation) and M_{map} is the total persistent checkpoint size on disk (all files written by the system's save routine, including network weights, feature stores, and codebooks). This dimensionless ratio decomposes runtime cost into the map itself and the *computational scaffolding*: optimizer state, gradient buffers, rendering caches, vocabulary data, and allocator overhead. Low α indicates that map size predicts deployment cost; high α indicates hidden runtime expenses. Note that α should always be interpreted alongside absolute M_{peak} and M_{map}; Table 3 reports both quantities.

Important caveat: We distinguish α_{CPU} (CPU RSS) from α_{GPU} (GPU device allocation). These are not directly comparable: CPU RSS includes libraries and OS overhead; GPU allocation captures optimizer state and CUDA caches

Table 3. Unified efficiency analysis across spatial memory paradigms. Overhead factor $\alpha = M_{\text{peak}}/M_{\text{map}}$; α_{CPU} denotes CPU RSS, α_{GPU} denotes GPU allocation (see Section 4). **Cross-benchmark caveat:** sparse systems are evaluated on EuRoC (real-world, stereo-inertial) and neural systems on Replica (synthetic, RGB-D); ATE and memory values are *not* directly comparable across horizontal dividers. Within-benchmark comparisons are valid; cross-paradigm conclusions should be drawn from α and scaling behavior. Per-value provenance notes are in Tables 1 and 2.

System	Paradigm	ATE (cm)	PSNR (dB)	Map (MB)	Peak (MB)	FPS	α
Sparse — evaluated on EuRoC [4] (real-world, stereo-inertial)							
ORB-SLAM3 [5]	Sparse	3.5	—	55	220†	30	4.0†
VINS-Mono [52]	Sparse (VI)	10.6‡	—	40	—	20	—
Basalt [69]	Sparse (VI)	8.8§	—	35§	120§	30§	$\sim$3.4§
Learned Flow — evaluated on EuRoC / TUM RGB-D (real-world)							
DROID-SLAM [64]	Learned Flow	2.0*	—	—**	11,000**	6	—
Neural — evaluated on Replica [61] (synthetic, RGB-D)							
iMAP [63]	NeRF	3.12***	22.1	1	—	3	—
NICE-SLAM [86]	NeRF	1.06	24.4	47$^{\diamond}$	10,082$^{\diamond}$	2	215$^{\diamond}$
Co-SLAM [71]	NeRF	1.00	30.2	8$^{\diamond}$	1,258$^{\diamond}$	16	157$^{\diamond}$
Point-SLAM [56]	NeRF	0.52	35.2	2,865$^{\diamond}$	6,563$^{\diamond}$	1	2.3$^{\diamond}$
SplaTAM [29]	3DGS	0.36	34.1	254$^{\diamond}$	14,024$^{\diamond}$	—	55$^{\diamond}$
MonoGS [42]	3DGS	0.58	38.9	90	—	3	—
GS-SLAM [77]	3DGS	0.50	34.3	198	—	8	—
SGS-SLAM [39]	3DGS (Sem.)	0.41	34.7	254$^{\diamond}$	40,330$^{\diamond}$	2	159$^{\diamond}$
Scale Reference — evaluated on custom outdoor sequences (not directly comparable)							
GigaSLAM [10]	3DGS (Hier.)	—$^{\|}$	$\sim$24$^{|}$	—$^{\|}$	—	3	—
Dense / Hierarchical — hardware-dependent							
OctoMap [23]	Dense (Octree)	—	—	45	—	10	—
Hydra [26]	Scene Graph	—	—	48$^{\#}$	—	5	—

‡ Average RMS ATE over MH01–MH05 (monocular-inertial); reproduced by [5], Table II.
§ Survey estimates from profiling; ATE computed from Table I of [69] (stereo VIO, MH01–MH05).
* Approximate average ATE on EuRoC (monocular) from [64] Table 2.
** DROID-SLAM does not produce a persistent map file; M_{peak} is GPU memory (requires $\geq$11 GB per [64]). α is not applicable.
† α_{CPU}: CPU RSS measured on Intel i7-10700 (five runs, stereo-inertial).
$^{\diamond}$ α_{GPU}: independently measured via `nvidia-smi` at 1 Hz on Replica/room0; NVIDIA A100-SXM4-80GB. Baseline GPU memory subtracted. Map sizes from saved checkpoint files on disk. Checkpoint discrepancies (ours vs. prior literature): Co-SLAM 8 vs. 32 MB, NICE-SLAM 47 vs. 235 MB, Point-SLAM 2,865 vs. 80 MB, SplaTAM 254 vs. 85 MB, SGS-SLAM 254 vs. 92 MB. See Section 4.
*** Room0 only from MonoGS [42] Table 2; avg. across 8 Replica scenes is 2.58 cm.
$^{\|}$ GigaSLAM reports km-scale outdoor mapping; ATE is on custom sequences not directly comparable to Replica. PSNR is avg. on KITTI from Table 3. Map size not separately reported (GPU memory is 8–22 GB including runtime overhead).
$^{\#}$ Graph abstraction layer only; excludes the underlying Kimera metric-semantic mesh, which adds additional memory.

but not the host process. Table 3 labels each explicitly. Within GPU methods alone, α_{GPU} spans from 2.3 (Point-SLAM, map-dominated) to 215 (NICE-SLAM, overhead-dominated).

Few systems report M_{peak} directly. Our independent profiling reveals that α varies by two orders of magnitude: from ORB-SLAM3 ($\alpha_{\text{CPU}} = 4$, RSS on Intel i7-10700) and Point-SLAM ($\alpha_{\text{GPU}} = 2.3$, A100-SXM4-80GB) at one extreme, to NICE-SLAM ($\alpha_{\text{GPU}} = 215$) at the other, meaning a 47 MB saved map consumes over 10 GB of GPU memory. Critically, Point-SLAM's low α does not imply efficiency, since its 2.9 GB checkpoint already dominates the 6.6 GB peak, whereas Co-SLAM's low M_{map} (8 MB) masks 1.3 GB of runtime cost. We advocate that future work routinely report M_{peak} alongside M_{map}.

Architectural drivers of α. Three design choices largely determine α: (1) *representation compactness vs. runtime scaffolding*: compact implicit methods (Co-SLAM, NICE-SLAM) store only network weights or hash tables on disk, but require optimizer state (Adam [32] stores two momentum buffers per parameter, $\sim 3\times$ the model size), activation maps, and CUDA memory pool reservations at runtime, producing $\alpha \gg 1$; (2) *map-dominant architectures*: Point-SLAM stores per-point neural features directly in the checkpoint (2.9 GB), so the map itself dominates runtime cost and $\alpha \approx 2.3$, but reducing α alone does not guarantee efficiency since Point-SLAM has the highest absolute M_{peak} after SGS-SLAM; (3) *accumulation without pruning*: SplaTAM and SGS-SLAM continuously add Gaussians without merging or culling, amplifying α to 55–159 as rendering buffers scale with Gaussian count. In summary, low α can arise from genuinely efficient runtime (sparse SLAM) or from front-loading cost into the map (Point-SLAM); high α signals that runtime scaffolding dwarfs the persistent representation.

Measurement methodology. We profiled five neural SLAM systems on Replica/room0 (NVIDIA A100-SXM4-80GB): Co-SLAM [71], NICE-SLAM [86], Point-SLAM [56], SplaTAM [29], and SGS-SLAM [39]. Protocol: peak GPU memory sampled at 1 Hz via `nvidia-smi`,

baseline subtracted, map size from saved checkpoint. Checkpoint sizes revealed systematic discrepancies with literature in every case (ours vs. literature): Co-SLAM 8 vs. 32 MB, NICE-SLAM 47 vs. 235 MB, Point-SLAM 2,865 vs. 80 MB, SplaTAM 254 vs. 85 MB, SGS-SLAM 254 vs. 92 MB. The 3DGS checkpoints are $\sim 3\times$ larger because saved Gaussian parameters exceed typically reported counts. Values marked $\diamond$ in Table 3 are independently measured; Fig. 4 shows memory time series.

Hardware sensitivity. Much of the high α_{GPU} is training-time scaffolding (Adam [32] $\sim 3\times$ overhead); inference-only deployment could substantially reduce α_{GPU}, though no surveyed system profiles this. α is also hardware-dependent: Co-SLAM shows ~ 3.2 GB peak on RTX 3090 (literature) vs. 1.3 GB on A100 (measured), due to GPU-specific CUDA allocator behavior. We recommend that future α measurements report exact GPU model, baseline memory, and sampling method.

4.2. Cross-Paradigm Comparison

Table 3 presents the unified comparison. We omit composite efficiency ratios as they produce misleading rankings across benchmarks; the Pareto visualization (Fig. 3) preserves both dimensions. Within-benchmark comparisons are valid; cross-paradigm conclusions should be drawn from α and scaling behavior rather than absolute accuracy (see Table 3 caption).

4.3. Pareto Analysis

Fig. 3 visualizes the tradeoff landscape with explicit benchmark separation (EuRoC left, Replica right); these benchmarks are not directly comparable, hence both tables and figure separate them. Learned-flow systems (DROID-SLAM, DPVO) are excluded as they lack persistent maps. The dashed Pareto front on Replica traces five non-dominated points: iMAP→Co-SLAM→MonoGS→GS-SLAM→SplaTAM. The largest gain is iMAP→Co-SLAM ($3\times$ ATE improvement at $8\times$ map cost); the three 3DGS systems then cluster in the 90–254 MB range with diminishing returns (MonoGS map size is a survey estimate; GS-SLAM from their Table 4).

On EuRoC (Fig. 3, left), ORB-SLAM3 achieves the best ATE (3.5 cm) at 55 MB; Basalt trades accuracy for a smaller map (35 MB). On Replica (right), 3DGS systems cluster favorably (ATE ≤ 0.58 cm, 90–254 MB), with SplaTAM achieving the best ATE (0.36 cm) at $\alpha_{\text{GPU}} = 55$. Point-SLAM's true checkpoint (2.9 GB) places it far from the Pareto front, invisible under the 80 MB survey estimate. The binding constraint differs by paradigm: runtime overhead for compact implicit methods ($\alpha_{\text{GPU}} = 157$, Co-SLAM) vs. map size for dense-point methods ($\alpha = 2.3$, Point-SLAM).

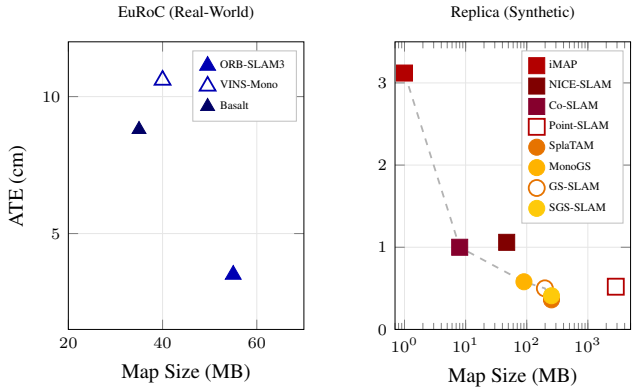

Figure 3. Accuracy vs. map size (M_{map}) by benchmark. **Left:** Sparse (EuRoC, linear scale). **Right:** Neural (Replica, log scale); dashed = Pareto front. Lower-left is better. Point-SLAM's measured 2.9 GB checkpoint (vs. 80 MB survey estimate) moves it far right, off the Pareto front. Plotting M_{peak} instead would further shift high-α systems rightward.

5. Evaluation and Practical Guidance

The overhead factor α captures an important dimension, but a single ratio is insufficient for characterizing spatial memory efficiency. We propose four complementary metrics that no surveyed system currently reports:

1. **Memory growth rate** (dM/dt): distinguishes bounded systems (RTAB-Map, hash-based NeRF) from unbounded ones (vanilla 3DGS); low final M_{map} but high dM/dt leads to out-of-memory failures on longer missions.
2. **Query latency**: varies by orders of magnitude ($O(1)$ hash lookups to full NeRF inference) yet is never benchmarked comparatively.
3. **Memory–completeness curve**: F1 score vs. cumulative map size, revealing diminishing returns in reconstruction quality.
4. **Throughput degradation**: FPS as map size approaches the memory ceiling; short benchmarks (<5 min) never stress this.

We demonstrate memory growth rate via independent profiling (Fig. 4): GPU memory sampled at 1 Hz on Replica/room0, showing that Co-SLAM is bounded, SplaTAM grows linearly, and NICE-SLAM oscillates with periodic global passes. The remaining three metrics are not yet benchmarked in any surveyed system; we formalize them here as concrete targets for future evaluation infrastructure.

Proposed evaluation protocol. We recommend that future work: (1) report M_{map}, M_{peak}, and α with hardware specs; (2) evaluate on sequences $\geq$30 min to stress memory management; (3) separate training-time from inference-time α; (4) profile on both discrete-GPU and embedded platforms. Current benchmarks (EuRoC [4], Replica [61],

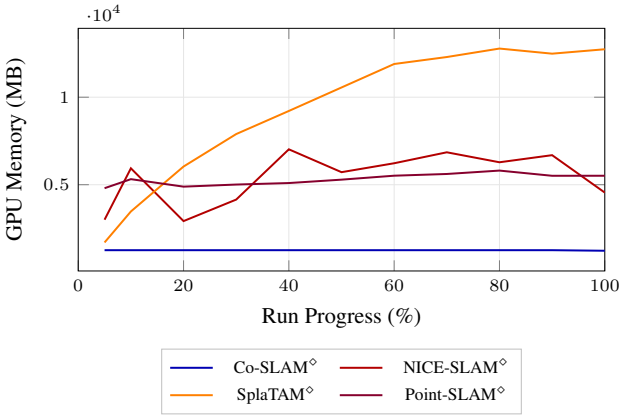

Figure 4. Runtime GPU memory on Replica/room0 (1 Hz sampling, A100-SXM4-80GB, baseline subtracted). All four systems independently measured (◇). Co-SLAM stays flat at ∼1.3 GB (bounded hash). NICE-SLAM oscillates between 3–7 GB due to periodic global mapping passes (absolute peak 10.1 GB, Table 3). Point-SLAM stabilizes at 5–6 GB after initialization. SplaTAM grows monotonically to ∼12.7 GB as Gaussians accumulate (peak 14.0 GB).

Algorithm 1: α-Aware Memory Budgeting

Input: M_{budget}: available RAM/VRAM after OS and drivers
Output: Feasible map size and representation choice
Look up α for candidate paradigm (Table 3);
$M_{\text{map}}^{\max} \leftarrow M_{\text{budget}} \,/\, \alpha$; // Eq. 3
if $M_{map}^{\max} \geq M_{target}$ **then**
  Select representation from Table 4;
else
  Apply compression (10–25× [17, 37]), adopt submaps/streaming, or choose sparser representation;
end

TUM RGB-D [62]) cover minutes at room-scale; long-duration benchmarks recording $M(t)$ and FPS(t) would be a high-impact contribution.

5.1. Deployment Budgeting

We distill our analysis into a systematic procedure. Given a target platform, Algorithm 1 computes the largest map the hardware can support; Table 4 then narrows the paradigm choice. The key insight: work *backward* from deployment memory constraints using α (Table 3), not forward from map size benchmarks:

$$M_{\text{map}}^{\max} = \frac{M_{\text{budget}}}{\alpha} \tag{3}$$

Example: Jetson Orin NX (16 GB) with high-α neural SLAM ($\alpha_{\text{GPU}} \in [55, 215]$): $M_{\text{map}}^{\max} \approx$ 75–290 MB. With sparse SLAM ($\alpha_{\text{CPU}} \approx 4$): ∼4 GB. These α values are room-scale; building-scale deployments may differ.

Table 4. Representation selection guide by primary constraint.

Constraint	Recommended	α range	Type	Avoid
CPU-only (<8 GB)	Sparse, Octree	3–5[a]	CPU	Neural
Embedded GPU (<16 GB)	Sparse, SG	4–10*	CPU	Raw 3DGS
Dense geometry	TSDF, 3DGS	∼55[b]	GPU	Sparse only
Photo rendering	3DGS, NeRF	2–215[c]	GPU	Occupancy
Multi-hour	Submaps, Stream	varies	—	Monolithic
Semantic	SG, VLM	4–10*	CPU	Geom. only

[a] Basalt ($\alpha_{\text{CPU}} \approx 3.4$) to ORB-SLAM3 ($\alpha_{\text{CPU}} = 4.0$).
[b] SplaTAM ($\alpha_{\text{GPU}} = 55$, measured); hardware-dependent (Sec. 4).
[c] Point-SLAM (2.3, measured) to NICE-SLAM (215, measured). SplaTAM (55), Co-SLAM (157), SGS-SLAM (159) in range.
* Graph layer only; total α with metric backend is higher.
Type: CPU RSS or GPU allocation (Sec. 4). SG = Scene Graph.

Table 5. Memory dynamics taxonomy. Consol. = consolidation, Hier = hierarchical, Mono = monolithic. ΔM = reported memory reduction. α from Table 3 where available; note that forgetting reduces M_{map} but not α.

System	Update	Forget	Part.	α	ΔM
ORB-SLAM3 [5]	Incr+BA	Culling	Sub	4	—[a]
RTAB-Map [35]	Incr	WM→LTM	Sub	—	Bound.[b]
DSO [15]	Window	Temporal	Mono	—	—
MS-SLAM [82]	Window	Sparsif.	Mono	—	>70%[c]
Co-SLAM [71]	Incr	None	Mono	157	—
GigaSLAM [10]	Incr	LOD	Hier	—	LOD[d]
MemGS [2]	Incr	Merging	Mono	—	—[e]
BioSLAM [78]	Gated	Consol.	Hier	—	+24%[f]
Hydra [26]	Incr	None	Hier	—	—

[a] Culling impact not quantified in the original paper.
[b] Active memory bounded by WM size parameter regardless of session length.
[c] >70% reduction in peak memory increase vs. ORB-SLAM3 [82].
[d] Level-of-detail rendering reduces active Gaussians; total map persists on disk.
[e] Merging reduces Gaussian count but impact not separately quantified.
[f] +24% weighted recall vs. generative replay; memory impact not reported.

6. Memory Dynamics

Even compact representations accumulate data without active management. We organize memory dynamics along three dimensions (Table 5): *update policy* (how new observations are incorporated), *forgetting rule* (whether and how old data is discarded), and *partitioning* (monolithic vs. hierarchical memory organization). A key distinction: forgetting reduces M_{map} but not α: Co-SLAM has no forgetting mechanism yet achieves bounded M_{map} via its fixed-size hash table, while still exhibiting $\alpha = 157$ because runtime scaffolding is unchanged. Reducing α requires *architectural* changes (inference-only deployment, gradient checkpointing, mixed-precision training) that shrink the scaffolding itself.

Our profiling (Fig. 4) maps directly onto this taxonomy: Co-SLAM's flat trace reflects its pre-allocated hash (bounded $M(t)$), NICE-SLAM oscillates from periodic mapping spikes, SplaTAM grows monotonically without pruning (cf. MemGS [2]), and Point-SLAM stabilizes after initialization. All four neural systems lack explicit forgetting. Classical methods are more disciplined: ORB-SLAM3 culls keyframes, RTAB-Map transfers to disk,

DSO [15] applies temporal windowing, and BioSLAM [78] uses gated consolidation (Table 5).

Multi-robot systems add a distribution dimension: centralized [58, 59] vs. decentralized [36] architectures multiply M_{map}, and collaborative 3DGS [66, 76, 79] is emerging with unknown merging cost [48].

7. Open Challenges

Sparse systems exhibit *explicit* failure modes (tracking loss triggers relocalization [52]), whereas neural methods degrade *silently* (hash saturation [71]) or *catastrophically* (out-of-memory termination [2]).

City-scale neural SLAM. GigaSLAM [10] demonstrates kilometer-scale 3DGS, but true city-scale operation ($\sim 10^7$ m^3, $>$100 h, $<$30 W) remains open. Even at $15\times$ compression [17], city-scale maps reach multi-gigabyte range without LOD management. DiskChunGS [18] addresses streaming via spatial chunking, but the embedded gap remains severe ($\sim$0.01 FPS on Jetson AGX Orin [20]). Cloud-edge splitting is established for feature-based SLAM [3, 50, 58] but unexplored for 3DGS.

Lifelong map maintenance. Neural SLAM faces catastrophic forgetting [33]: new observations overwrite weights encoding earlier regions. Mitigations include elastic weight consolidation [33], CL-Splats [1], VBGS [70], GaussianUpdate [80], and WildGS-SLAM [83], yet all target single-session neural map updates. BioSLAM [78] addresses multi-session place recognition but not map-level continual learning for neural representations. Information-theoretic criteria for *when to forget* (e.g., retaining only observations that reduce map entropy) are largely unexplored.

Uncertainty quantification. Neural representations return confident outputs for unobserved regions, a critical failure mode for safety-critical planning. Stochastic approaches multiply inference cost, and storing per-element variance doubles map storage. Moment-based uncertainty [16] propagates moments through rendering without additional training, though memory overhead in full SLAM is unverified.

Foundation model features. CLIP features (d=512, 32-bit) cost $\sim$2 GB per million points, adding a memory layer *on top of* the geometric map. Compression strategies include per-segment storage (HOV-SG [74], 75% reduction), task-driven clustering (Clio [41]), and product quantization (Dr. Splat [31], $\sim$6% ratio). Even compressed, language features add $\sim$50% per-Gaussian overhead. No surveyed paper reports α for semantic systems.

Hardware–algorithm co-design. Unified memory architectures (NVIDIA Jetson Orin, Apple Silicon) may reduce α by eliminating duplicate CPU–GPU buffers. Custom ASICs (REACT3D [72]: $12\times$ throughput) and software optimization [47] (14 FPS on Orin NX) narrow the gap but neither achieves full-resolution real-time on stock hardware.

Toward scalable solutions. No single technique permanently solves memory scaling, because data complexity grows with environment size and mission duration. However, three complementary strategies converge toward *bounded-memory operation*: (1) *hierarchical streaming* (cf. GigaSLAM [10], DiskChunGS [18]), bounding runtime memory at the cost of disk I/O; (2) *information-theoretic forgetting*, achieving $O(\log t)$ growth in principle; and (3) *hardware–algorithm co-design*, reducing α structurally. A system combining all three would approach indefinite operation under fixed memory, where cost depends on the *active working area* rather than total mission history. No existing system achieves this.

8. Conclusion

This survey traced spatial memory evolution across over three decades and 88 references around a central finding: *map size alone is an unreliable proxy for deployment cost*. Our overhead factor α spans two orders of magnitude ($\alpha_{\text{GPU}} = 2.3$–$215$), revealing that compact implicit methods pay orders-of-magnitude runtime overhead, dense neural point methods embed cost in the map itself, and 3DGS methods inherit high α from unbounded accumulation. No single paradigm dominates (Fig. 3), and memory growth dynamics (Fig. 4) reveal failure modes invisible in static metrics.

Recommendations. (1) Adopt α-aware reporting (M_{peak} + hardware specs alongside M_{map}); (2) develop long-duration benchmarks ($\geq$30 min) that record $M(t)$ and stress-test growth to failure; (3) establish inference-time profiling protocols separating training overhead from deployment cost.

Limitations. Our profiling covers five neural SLAM systems on a single Replica scene using one GPU (A100); α values may differ on other scenes, datasets, or hardware. Only training-time α is measured; inference-only α remains unprofiled. Benchmark heterogeneity constrains cross-paradigm comparisons to α and scaling behavior.

This survey is designed to be actionable: the taxonomy (Fig. 2) maps the landscape, Table 4 matches paradigms to hardware, and Algorithm 1 computes maximum feasible map size from the memory budget and α. As neural SLAM matures toward deployment, we expect α-aware design to become standard practice.

References

[1] Jan Ackermann, Jonas Kulhanek, Shengqu Cai, Haofei Xu, Marc Pollefeys, Gordon Wetzstein, Leonidas Guibas, and Songyou Peng. CL-Splats: Continual learning of Gaussian Splatting with local optimization. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

[2] Yinlong Bai, Hongxin Zhang, Sheng Zhong, Junkai Niu, Hai Li, Yijia He, and Yi Zhou. MemGS: Memory-efficient Gaussian splatting for real-time SLAM. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11097–11103, 2025.

[3] Ali J. Ben Ali, Zakieh Sadat Hashemifar, and Karthik Dantu. Edge-SLAM: Edge-assisted visual simultaneous localization and mapping. In *Proceedings of the 18th ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 325–337, 2020.

[4] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W. Achtelik, and Roland Siegwart. The EuRoC micro aerial vehicle datasets. *International Journal of Robotics Research*, 35(10): 1157–1163, 2016.

[5] Carlos Campos, Richard Elvira, Juan J. Gómez Rodríguez, José M.M. Montiel, and Juan D. Tardós. ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM. *IEEE Transactions on Robotics*, 37(6): 1874–1890, 2021.

[6] Guikun Chen and Wenguan Wang. A survey on 3D Gaussian Splatting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[7] Timothy Chen, Ola Shorinwa, Joseph Bruno, Aiden Swann, Javier Yu, Weijia Zeng, Keiko Nagami, Philip Dames, and Mac Schwager. Splat-Nav: Safe real-time robot navigation in Gaussian splatting maps. *IEEE Transactions on Robotics*, 41, 2025.

[8] Mark Cummins and Paul Newman. FAB-MAP: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 27(6):647–665, 2008.

[9] Andrew J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1403–1410, 2003.

[10] Kai Deng, Jian Yang, Shenlong Wang, and Jin Xie. GigaSLAM: Large-scale monocular SLAM with hierarchical Gaussian splats. In *ACM SIGGRAPH Asia Conference Papers*, 2025.

[11] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperPoint: Self-supervised interest point detection and description. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 224–236, 2018.

[12] Daniel Duberg and Patric Jensfelt. UFOMap: An efficient probabilistic 3D mapping framework that embraces the unknown. *IEEE Robotics and Automation Letters*, 5(4): 6411–6418, 2020.

[13] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989.

[14] Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *European Conference on Computer Vision (ECCV)*, pages 834–849, 2014.

[15] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):611–625, 2018.

[16] Parker Ewen, Hao Chen, Seth Isaacson, Joey Wilson, Katherine A. Skinner, and Ram Vasudevan. These magic moments: Differentiable uncertainty quantification of radiance field models. *arXiv preprint arXiv:2503.14665*, 2025.

[17] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. LightGaussian: Unbounded 3D Gaussian compression with $15\times$ reduction and $200+$ FPS. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[18] Casimir Feldmann, Maximum Wilder-Smith, Vaishakh Patil, Michael Oechsle, Michael Niemeyer, Keisuke Tateno, and Marco Hutter. DiskChunGS: Large-scale 3D Gaussian SLAM through chunk-based memory management. *IEEE Robotics and Automation Letters*, 11(4):5009–5016, 2026.

[19] Samir Yitzhak Gadre, Mitchell Wortsman, Gabriel Ilharco, Ludwig Schmidt, and Shuran Song. CoWs on pasture: Baselines and benchmarks for language-driven zero-shot object navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 23171–23181, 2023.

[20] Calvin Galagain, Martyna Poreba, and François Goulette. Is semantic SLAM ready for embedded systems? A comparative survey. *arXiv preprint arXiv:2505.12384*, 2025.

[21] Dasong Gao, Peter Zhi Xuan Li, Vivienne Sze, and Sertac Karaman. GEVO: Memory-efficient monocular visual odometry using Gaussians. *IEEE Robotics and Automation Letters*, 2025.

[22] Seongbo Ha, Jiung Yeon, and Hyeonwoo Yu. RGBD GS-ICP SLAM. In *European Conference on Computer Vision (ECCV)*, 2024.

[23] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, 2013.

[24] Chenguang Huang, Oier Mees, Andy Zeng, and Wolfram Burgard. Visual language maps for robot navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 10608–10615, 2023.

[25] Huajian Huang, Longwei Li, Hui Cheng, and Sai-Kit Yeung. Photo-SLAM: Real-time simultaneous localization and photorealistic mapping for monocular, stereo, and RGB-D cameras. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[26] Nathan Hughes, Yun Yang, and Luca Carlone. Hydra: A real-time spatial perception system for 3D scene graph construction and optimization. In *Proceedings of Robotics: Science and Systems (RSS)*, 2022.

[27] Mohammad Mahdi Johari, Camilla Carta, and François Fleuret. ESLAM: Efficient dense SLAM system based on hybrid representation of signed distance fields. In *IEEE/CVF*

Conference on Computer Vision and Pattern Recognition (CVPR), pages 17408–17419, 2023.

[28] Saimouli Katragadda, Cho-Ying Wu, Yuliang Guo, Xinyu Huang, Guoquan Huang, and Liu Ren. Online language splatting. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

[29] Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. SplaTAM: Splat, track & map 3D Gaussians for dense RGB-D SLAM. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21357–21366, 2024.

[30] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for real-time radiance field rendering. *ACM Transactions on Graphics (TOG)*, 42(4):1–14, 2023.

[31] Jun-Seong Kim, GeonU Kim, Yu-Ji Kim, Yu-Chiang Frank Wang, Jaesung Choe, and Tae-Hyun Oh. Dr. Splat: Directly referring 3D Gaussian Splatting via direct language embedding registration. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. Highlight.

[32] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[33] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[34] Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. In *IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 225–234, 2007.

[35] Mathieu Labbé and François Michaud. RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics*, 36(2):416–446, 2019.

[36] Pierre-Yves Lajoie and Giovanni Beltrame. Swarm-SLAM: Sparse decentralized collaborative simultaneous localization and mapping framework for multi-robot systems. *IEEE Robotics and Automation Letters*, 9(1):475–482, 2024.

[37] Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3D Gaussian representation for radiance field. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21719–21728, 2024.

[38] Xiaohan Lei, Min Wang, Wengang Zhou, and Houqiang Li. GaussNav: Gaussian splatting for visual navigation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47:4108–4121, 2025.

[39] Mingrui Li, Shuhong Liu, Heng Zhou, Guohao Zhu, Na Cheng, Tianchen Deng, and Hongyu Wang. SGS-SLAM: Semantic Gaussian splatting for neural dense SLAM. In *European Conference on Computer Vision (ECCV)*, pages 163–179. Springer, 2024.

[40] Joel Loo, Zhanxin Wu, and David Hsu. Open scene graphs for open-world object-goal navigation. In *ICRA Workshop on Vision-Language Models for Navigation and Manipulation (VLMNM)*, 2024.

[41] Dominic Maggio, Yun Chang, Nathan Hughes, Matthew Trang, Dan Griffith, Carlyn Dougherty, Eric Cristofalo, Lukas Schmid, and Luca Carlone. Clio: Real-time task-driven open-set 3D scene graphs. *IEEE Robotics and Automation Letters*, 9(10):8921–8928, 2024.

[42] Hidenobu Matsuki, Riku Murai, Paul H. J. Kelly, and Andrew J. Davison. Gaussian splatting SLAM. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[43] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision (ECCV)*, pages 405–421, 2020.

[44] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (TOG)*, 41(4):102:1–102:15, 2022.

[45] Raul Mur-Artal and Juan D. Tardós. ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[46] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D. Tardós. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[47] Abdoullah Ndoye, Amaury Nègre, Nicolas Marchand, and Franck Ruffier. VIGS-Fusion: Fast Gaussian splatting SLAM processed onboard a small quadrotor. In *IEEE International Conference on Advanced Robotics (ICAR)*, 2025.

[48] Phuc Nguyen Xuan, Thanh Nguyen Canh, Huu-Hung Nguyen, Nak Young Chong, and Xiem HoangVan. A survey on collaborative SLAM with 3D Gaussian splatting. *arXiv preprint arXiv:2510.23988*, 2025.

[49] Helen Oleynikova, Zachary Taylor, Marius Fehr, Roland Siegwart, and Juan Nieto. Voxblox: Incremental 3D Euclidean signed distance fields for on-board MAV planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1366–1373, 2017.

[50] Manthan Patel, Marco Karrer, Philipp Banninger, and Margarita Chli. COVINS-G: A generic back-end for collaborative visual-inertial SLAM. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 8549–8555, 2023.

[51] Zhexi Peng, Tianjia Shao, Yong Liu, Jingke Zhou, Yin Yang, Jingdong Wang, and Kun Zhou. RTG-SLAM: Real-time 3D reconstruction at scale using Gaussian Splatting. In *ACM SIGGRAPH Conference Papers*, 2024.

[52] Tong Qin, Peiliang Li, and Shaojie Shen. VINS-Mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.

[53] Abhinav Rajvanshi, Karan Sikka, Xiao Lin, Bhoram Lee, Han-Pang Chiu, and Alvaro Velasquez. SayNav: Grounding

large language models for dynamic planning to navigation in new environments. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pages 464–474, 2024.

[54] Antoni Rosinol, Marcus Abate, Yun Chang, and Luca Carlone. Kimera: an open-source library for real-time metric-semantic localization and mapping. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1689–1696, 2020.

[55] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2564–2571, 2011.

[56] Erik Sandström, Yue Li, Luc Van Gool, and Martin R. Oswald. Point-SLAM: Dense neural point cloud-based SLAM. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 18433–18444, 2023.

[57] Erik Sandström, Ganlin Zhang, Keisuke Tateno, Michael Oechsle, Michael Niemeyer, Youmin Zhang, Manthan Patel, Luc Van Gool, Martin Oswald, and Federico Tombari. Splat-SLAM: Globally optimized RGB-only SLAM with 3D Gaussians. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1686–1697, 2025.

[58] Patrik Schmuck and Margarita Chli. CCM-SLAM: Robust and efficient centralized collaborative monocular SLAM for robotic teams. *Journal of Field Robotics*, 36(4):763–781, 2019.

[59] Patrik Schmuck, Thomas Ziegler, Marco Karrer, Jonathan Perraudin, and Margarita Chli. COVINS: Visual-inertial SLAM for centralized collaboration. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pages 171–176, 2021.

[60] Dhruv Shah, Błażej Osiński, Brian Ichter, and Sergey Levine. LM-Nav: Robotic navigation with large pre-trained models of language, vision, and action. In *Proceedings of the 6th Conference on Robot Learning (CoRL)*, pages 492–504, 2022.

[61] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.

[62] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 573–580, 2012.

[63] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J. Davison. iMAP: Implicit mapping and positioning in real-time. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6229–6238, 2021.

[64] Zachary Teed and Jia Deng. DROID-SLAM: Deep visual SLAM for monocular, stereo, and RGB-D cameras. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 16558–16569, 2021.

[65] Zachary Teed, Lahav Lipson, and Jia Deng. Deep patch visual odometry. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[66] Annika Thomas, Aneesa Sonawalla, Alex Rose, and Jonathan P. How. GRAND-SLAM: Local optimization for globally consistent large-scale multi-agent Gaussian SLAM. *IEEE Robotics and Automation Letters*, 10:13129–13136, 2025.

[67] Yulun Tian, Yun Chang, Fernando Herrera Arias, Carlos Nieto-Granda, Jonathan P. How, and Luca Carlone. Kimera-Multi: Robust, distributed, dense metric-semantic SLAM for multi-robot systems. *IEEE Transactions on Robotics*, 38(4): 2022–2038, 2022.

[68] Fabio Tosi, Youmin Zhang, Ziren Gong, Erik Sandström, Stefano Mattoccia, Martin R. Oswald, and Matteo Poggi. How NeRFs and 3D Gaussian Splatting are reshaping SLAM: a survey. *arXiv preprint arXiv:2402.13255*, 2024.

[69] Vladyslav Usenko, Nikolaus Demmel, David Schubert, Jörg Stückler, and Daniel Cremers. Visual-inertial mapping with non-linear factor recovery. *IEEE Robotics and Automation Letters*, 5(2):422–429, 2020.

[70] Toon Van de Maele, Ozan Catal, Alexander Tschantz, Christopher L. Buckley, and Tim Verbelen. Variational Bayes Gaussian splatting, 2024.

[71] Hengyi Wang, Jingwen Wang, and Lourdes Agapito. Co-SLAM: Joint coordinate and sparse parametric encodings for neural real-time SLAM. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13293–13302, 2023.

[72] Hongyi Wang, Zhenhua Zhu, Tianchen Zhao, Yunfei Xiang, Zehao Wang, Jincheng Yu, Huazhong Yang, Yuan Xie, and Yu Wang. REACT3D: Real-time edge accelerator for incremental training in 3D Gaussian Splatting based SLAM systems. In *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2025.

[73] Tianci Wen, Zhiang Liu, and Yongchun Fang. SEGS-SLAM: Structure-enhanced 3D Gaussian splatting slam with appearance embedding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

[74] Abdelrhman Werby, Chenguang Huang, Martin Büchner, Abhinav Valada, and Wolfram Burgard. Hierarchical open-vocabulary 3D scene graphs for language-grounded robot navigation. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[75] Quanting Xie, So Yeon Min, Tianyi Zhang, Kedi Xu, Aarav Bajaj, Ruslan Salakhutdinov, Matthew Johnson-Roberson, and Yonatan Bisk. Embodied-RAG: General non-parametric embodied memory for retrieval and generation. *arXiv preprint arXiv:2409.18313*, 2024.

[76] Xiaohao Xu, Feng Xue, Shibo Zhao, Yike Pan, Sebastian Scherer, and Xiaonan Huang. MAC-Ego3D: Multi-agent Gaussian consensus for real-time collaborative ego-motion and photorealistic 3D reconstruction. In *Proceedings of*

the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 854–863, 2025.

[77] Chi Yan, Delin Qu, Dan Xu, Bin Zhao, Zhigang Wang, Dong Wang, and Xuelong Li. GS-SLAM: Dense visual SLAM with 3D Gaussian Splatting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19595–19604, 2024.

[78] Peng Yin, Abulikemu Abuduweili, Shiqi Zhao, Lingyun Xu, Changliu Liu, and Sebastian Scherer. BioSLAM: A bioinspired lifelong memory system for general place recognition. *IEEE Transactions on Robotics*, 39(6):4855–4874, 2023.

[79] Javier Yu, Timothy Chen, and Mac Schwager. HAMMER: Heterogeneous, multi-robot semantic gaussian splatting. *IEEE Robotics and Automation Letters*, 2025.

[80] Lin Zeng, Boming Zhao, Jiarui Hu, Xujie Shen, Ziqiang Dang, Hujun Bao, and Zhaopeng Cui. GaussianUpdate: Continual 3D Gaussian Splatting update for changing environments. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

[81] Chenyangguang Zhang, Alexandros Delitzas, Fangjinhua Wang, Ruida Zhang, Xiangyang Ji, Marc Pollefeys, and Francis Engelmann. Open-vocabulary functional 3D scene graphs for real-world indoor spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[82] Xiaoyu Zhang, Jinhu Dong, Yin Zhang, and Yun-Hui Liu. MS-SLAM: Memory-efficient visual SLAM with sliding window map sparsification. *Journal of Field Robotics*, 2024.

[83] Jianhao Zheng, Zihan Zhu, Valentin Bieri, Marc Pollefeys, Songyou Peng, and Iro Armeni. WildGS-SLAM: Monocular gaussian splatting slam in dynamic environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[84] Liyuan Zhu, Yue Li, Erik Sandström, Shengyu Huang, Konrad Schindler, and Iro Armeni. LoopSplat: Loop closure by registering 3D Gaussian splats. In *International Conference on 3D Vision (3DV)*, 2025.

[85] Siting Zhu, Guangming Wang, Hermann Blum, Jiuming Liu, Liang Song, Marc Pollefeys, and Hesheng Wang. SNI-SLAM: Semantic neural implicit SLAM. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21167–21177, 2024.

[86] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys. NICE-SLAM: Neural implicit scalable encoding for SLAM. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12786–12796, 2022.