Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.086341

REVIEW

Fusion-Oriented Deep Learning-Enhanced Visual SLAM: A Review

Xiruo Chen, Qi Ouyang*, Sihong Meng and Yuke Meng

Department of Automation, Chongqing University, Chongqing, China

*Corresponding Author: Qi Ouyang. Email: yangqi@cqu.edu.cn

Received: 28 May 2026; Accepted: 03 July 2026; Published: 13 August 2026

ABSTRACT: Visual simultaneous localization and mapping (VSLAM) is a key technology for mobile robotics, autonomous driving, and embodied intelligence, enabling self-localization, environment reconstruction, and scene understanding. Although conventional geometric methods have achieved notable success, their performance often degrades in challenging conditions, such as low-texture scenes, severe illumination changes, dynamic interference, and long-term environmental variations. Recent advances in deep learning have created new opportunities to improve VSLAM through stronger feature representations, learned priors, semantic perception, and emerging map representations. At the same time, the increasing adoption of learning-based modules has raised important questions about integration strategies, generalization, interpretability, and real-time deployment. This paper presents a systematic review of deep learning-enhanced VSLAM, with a particular focus on how learning models are incorporated into classical simultaneous localization and mapping (SLAM) pipelines and how they function within the overall system. To provide a unified perspective, existing methods are organized into five categories according to their fusion interfaces with geometric SLAM pipelines: observation-level interfaces, constraint/prior/weight-level interfaces, solver-level interfaces, representation-level interfaces, and system-level integration interfaces. Based on this taxonomy, representative approaches are comparatively analyzed for accuracy, robustness, efficiency, and deployability. In addition, this review summarizes common design principles, including geometric consistency constraints, error propagation characteristics, and typical failure modes, and further discusses open challenges and future directions such as lightweight deployment, cross-domain adaptation, dynamic map modeling, and long-term consistency maintenance. This review aims to provide a structured reference for the analysis, design, and deployment of learning-enhanced VSLAM systems.

KEYWORDS: Visual SLAM; deep learning; localization and mapping; autonomous navigation

1 Introduction

Visual simultaneous localization and mapping (VSLAM) constitutes a fundamental enabling technology for robot navigation, augmented reality, and autonomous driving, as it jointly estimates camera trajectories and reconstructs environmental maps from continuous visual observations. Despite the notable success of conventional geometric pipelines built upon hand-crafted features and iterative optimization, their performance remains highly susceptible to real-world challenges, including severe illumination variations, weak or repetitive textures, dynamic objects, and long-term drift, all of which substantially compromise system stability and practical reliability [1,2].

In recent years, deep learning has been increasingly integrated into VSLAM systems, thereby opening new possibilities for improving both perception quality and estimation robustness. On the observation side, learning-based feature matching, depth prediction, optical flow estimation, and semantic masking can provide more stable and informative cues for front-end processing [1,3]. On the optimization side, confidence

estimation, robust reweighting, and differentiable optimization layers can further influence the convergence properties and error tolerance of geometric solvers. Meanwhile, emerging map representations, such as neural implicit models and Gaussian-based maps, have significantly advanced dense reconstruction and scene visualization capabilities [1,4,5].

Visual simultaneous localization and mapping (VSLAM) estimates the pose of a moving camera in real time within an unknown environment while progressively reconstructing the surrounding map from sequential visual observations. Compared with LiDAR-based solutions, vision-based approaches offer distinct advantages in cost-effectiveness, compact hardware requirements, and the ability to capture rich semantic information, making them particularly attractive for weight- and power-constrained platforms such as uncrewed aerial vehicles and service robots [2,6].

Despite these advantages, visual observations are intrinsically vulnerable to a variety of degradations, including illumination changes, occlusions, motion blur, and texture loss. More importantly, such disturbances may not remain confined to the perception stage; instead, they can propagate through the entire estimation pipeline, from front-end observation to back-end optimization and subsequent feedback correction, thereby leading to cumulative performance deterioration. In this context, the introduction of deep learning aims to furnish geometric pipelines with more reliable observations, more controllable priors, and richer scene representations, ultimately improving the robustness and deployability of VSLAM systems in complex real-world scenarios [1,3,4].

As illustrated in Fig. 1, a typical VSLAM system comprises three major components: the front-end, the back-end, and the loop closure/relocalization module. The front-end is responsible for extracting repeatable and informative observations from image sequences, including key points, descriptors, dense depth maps, optical flow, and semantic masks, and for establishing inter-frame correspondences. The quality of these observations directly affects the observability and stability of the subsequent pose estimation process in the back end [2,6].

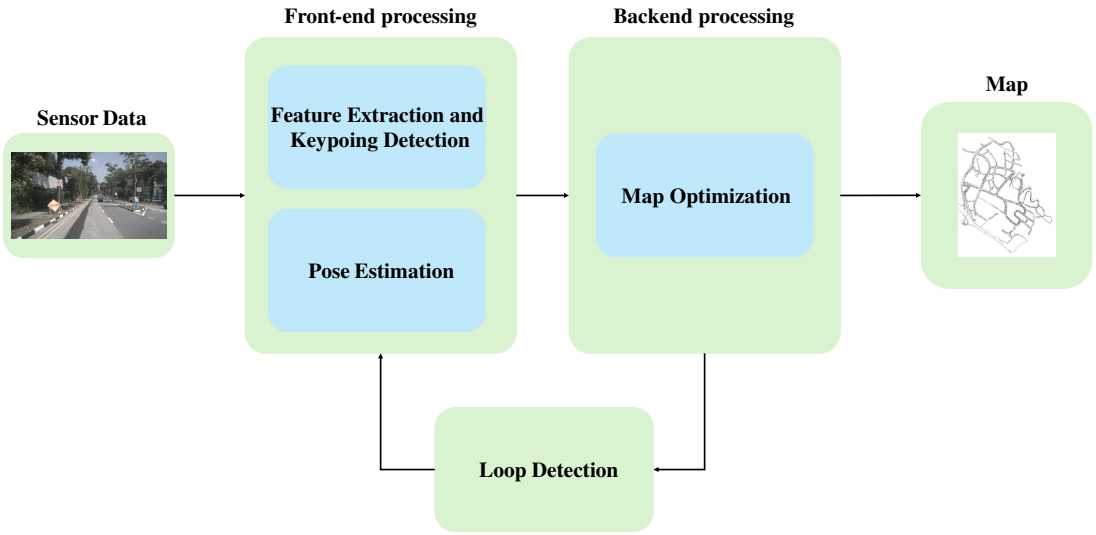

Figure 1: The typical visual SLAM system framework.

The back end is fundamentally built upon geometric constraints and performs joint estimation of camera trajectories and map structure through nonlinear optimization. When necessary, it further invokes loop closure detection and global consistency correction to suppress accumulated drift. However, as operating

time increases and scene scale expands, error propagation and computational complexity gradually become unavoidable bottlenecks that constrain overall system performance.

The mainstream paradigm of deep learning-enhanced VSLAM is not to replace the geometric pipeline entirely, but rather to strengthen it by providing more reliable inputs and more favorable optimization conditions at critical stages. Specifically, learning-based modules can enrich the observation space through more robust key points and feature matches, depth or optical flow estimation, and semantic segmentation outputs. They can also improve the optimization process by introducing confidence and uncertainty estimation, robust kernel parameter learning, and residual reweighting, thereby yielding more reasonable priors and weight distributions. Furthermore, optimization procedures such as Perspective-n-Point (PnP), bundle adjustment (BA), and pose graph optimization (PGO) can be embedded in neural networks in a differentiable manner, enabling end-to-end training. In parallel, emerging map representations, such as neural implicit representations and 3D Gaussian Splatting, further enhance the expressive power of scene modeling [1,3–5]. From the perspective of this integrated framework, the following sections of this paper will present a systematic analysis and comparative review of related methods.

Compared with existing review articles, this paper places greater emphasis on the fusion interfaces and functional mechanisms through which learning modules interact with geometric VSLAM pipelines. Existing surveys have provided valuable summaries of visual SLAM, deep learning-based SLAM, semantic SLAM, sensor configurations, neural network architectures, and application scenarios. These works help clarify the evolution from traditional geometric SLAM to learning-enhanced and semantic SLAM. However, as learning-based modules are increasingly embedded at different stages of the SLAM pipeline, a purely task-level or network-level summary is no longer sufficient to explain how learning affects geometric estimation. The same type of neural output may play different roles depending on whether it is used as an observation, a prior, a residual weight, a differentiable solver component, a map representation, or a system-level scheduling cue.

Therefore, this review adopts an interface-oriented perspective. Rather than following only the chronological transition from traditional VSLAM to semantic VSLAM, it focuses on where learning outputs enter the geometric pipeline and how they affect pose estimation, mapping, loop closure, uncertainty handling, and system deployment. On this basis, existing methods are organized into five fusion interfaces: observation-level interfaces, constraint/prior/weight-level interfaces, solver-level interfaces, representation-level interfaces, and system-level integration interfaces. This organization clarifies the relationship between learning modules and classical geometric components and supports the analysis of hybrid systems, failure modes, and deployment constraints.

Our review makes the following contributions to the state of the art:

- We propose a fusion-interface taxonomy that categorizes learning-enhanced VSLAM methods by integration interface, including observation-level, constraint/prior/weight-level, solver-level, representation-level, and system-level interfaces.
- We clarify the boundary rules for hybrid systems by distinguishing the dominant fusion interface from secondary cross-category attributes.
- We analyze representative methods with respect to their functional mechanisms, accuracy, robustness, efficiency, and deployability.
- We summarize key trade-offs and failure modes, including cross-domain bias, uncertainty miscalibration, computational burden, dynamic-scene interference, and long-term consistency degradation.

As shown in Fig. 2, the remainder of this paper is organized to address the aforementioned research gaps progressively, moving from background context to system-level analysis and, finally, to open challenges and

future directions. Section 2 briefly reviews commonly used sensors, evaluation metrics, and public datasets, establishing a unified basis for the comparative discussions that follow. As the core of the paper, Section 3 develops an interface-oriented framework centered on the interaction between learning outputs and geometric VSLAM pipelines. This organization clarifies how learned features, matches, depth, flow, semantic masks, uncertainty estimates, differentiable solvers, and neural maps affect the stages of visual SLAM. Section 4 discusses frontier directions, open challenges, and future research trends, including foundation models and vision-language integration, uncertainty calibration, real-time deployment, learned-component failure modes, dynamic map maintenance, and long-term system reliability.

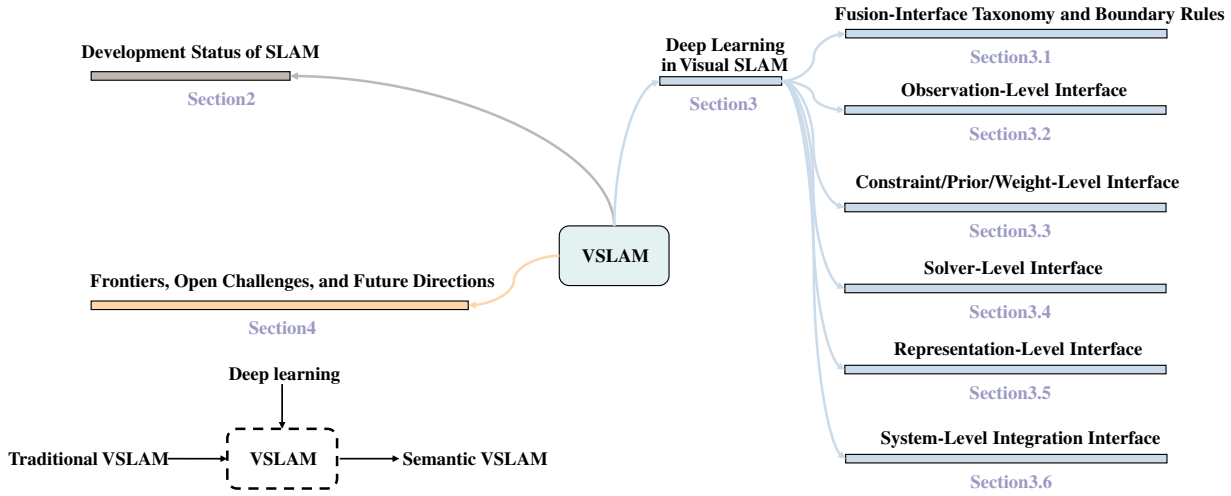

Figure 2: Schematic diagram of the structure of the remainder of this paper. This paper focuses on deep learning in VSLAM, as discussed in Section 3. We believe that the introduction of neural networks marks the beginning of semantic VSLAM. We classify approaches by fusion method, systematically compare their accuracy, robustness, and computational efficiency, and provide a summary and outlook.

2 The Development State of Visual SLAM

From a system-level perspective, the choice of sensor fundamentally determines the observability limits of a VSLAM system and shapes the primary interfaces for incorporating deep learning [1,2,4,6]. Monocular systems rely most heavily on learned depth and motion priors to compensate for the inherent lack of absolute scale. By contrast, stereo and RGB-D systems provide more stable scale observability, but their performance is more closely tied to robust correspondence estimation, dynamic object suppression, and dense scene representation [1,4,6]. Event cameras, panoramic cameras, and visual–inertial configurations are generally better suited to scenarios involving high-speed motion, wide fields of view, and texture-deficient environments [2,7,8]. Accordingly, the performance of learning-enhanced VSLAM should not be discussed independently of sensor configuration; rather, sensors, scene conditions, and algorithmic interfaces should be considered as a tightly coupled whole. The characteristics and major limitations of commonly used visual sensors are summarized in Table 1.

For deep learning-enhanced VSLAM, system context and evaluation criteria should not be treated as separate issues. Sensor configurations define the system's observability limits and operational constraints; scenario-specific challenges determine the extent to which learning-based modules can deliver performance gains; and evaluation protocols directly affect the interpretation of a method's effectiveness. Therefore, a fair and meaningful analysis of learning-enhanced VSLAM must consider these factors in an integrated manner. Table 2 summarizes several representative and widely used datasets for VSLAM evaluation.

Table 1: Cameras commonly used in VSLAM.

Cameras	Advantages	Disadvantages
Monocular 	Low-cost, lightweight	Unable to detect object depth
Stereo 	Capable of sensing depth at a low cost	Low computational efficiency
RGB-D 	Information-rich and computationally efficient	Susceptible to light interference, limited field of view, and high noise levels

Table 2: Typical public datasets and their main stress factors.

Dataset	Sensor	Environment	Data Volume	Evaluation Criteria
TUM RGB-D [9]	RGB-D (Kinect), IMU	Indoors, handheld camera	About 50 sequences	ATE, RPE
ICL-NUIM [10]	Stereo, RGB-D	Indoor, composite scene	4 scenes, thousands of frames	ATE, RMSE
EuRoC MAV [11]	Stereo, IMU	Indoors, drone flight	11 sequences	ATE, RMSE, Vicon True Value
KITTI Odometry [12]	Stereo, LiDAR, GPS/IMU	Outdoors, Autonomous Driving	22 sequences	ATE, KITTI Odometry Benchmark
Oxford RobotCar [13]	Multi-camera, LiDAR, GPS/INS	Outdoors, Autonomous Driving (All Seasons)	Tens of terabytes	Positioning accuracy, long-term robustness
New College [14]	Stereo, GPS/IMU	Outdoors, Robot Patrol	Several gigabytes	Path reconstruction accuracy, loopback testing
TartanAir [15]	RGB-D, IMU	Simulation, Multi-scenario/Motion	Several terabytes	ATE, VO/VIO robustness, deep learning training
7-Scenes [16]	RGB-D	Indoor, short-range positioning	Tens of thousands of frames	Positioning accuracy, loopback test
ScanNet [17]	RGB-D	Indoors, large-scale scanning	2.5 million frames	Reconstruction accuracy, semantic segmentation, SLAM evaluation

Note: RGB-D, Red, Green, Blue–Depth; IMU, Inertial Measurement Unit; INS, Inertial Navigation System; GPS, Global Positioning System.

As shown in Table 2, public datasets emphasize different stress factors. Indoor sequences are generally better suited for examining challenges related to weak textures, small-scale loop closure, and relocalization. In contrast, outdoor vehicular sequences are more effective at revealing difficulties caused by long-horizon drift, dynamic traffic participants, and temporal appearance variations. Simulated datasets are particularly valuable for controlled ablation studies, since illumination conditions, weather patterns, and motion modes can be explicitly manipulated by the system. Therefore, the evaluation of learning-enhanced VSLAM should, as far as possible, cover three groups of stress conditions, namely indoor vs. outdoor, static vs. dynamic, and in-domain vs. cross-domain settings, rather than relying solely on ATE performance in a single scenario to judge the merits of a given method.

Fig. 3 illustrates a steady increase in publications on visual SLAM and semantic SLAM in recent years. Meanwhile, Fig. 4 shows that research interests have expanded beyond conventional feature extraction and geometric optimization to include machine learning, semantic understanding, dynamic scene processing, and multi-sensor collaboration.

This transition implies that discussing algorithmic backgrounds or evaluation results in isolation is no longer sufficient to support a review's conclusions. Instead, research trends, scenario-specific challenges, and system-level evaluation should be interpreted within a unified analytical framework.

Against this background, this paper adopts a unified comparative framework to draw system-level conclusions. In terms of accuracy, beyond the commonly reported absolute trajectory error (ATE) and relative pose error (RPE), attention should also be paid to global consistency after loop closure to avoid obscuring long-range drift and local instability when relying solely on averaged error statistics. In terms of robustness, the evaluation should extend beyond the tracking success rate to include the number of tracking interruptions, the relocalization success rate, and the recovery time, thereby reflecting the system's ability to operate continuously under dynamic, occluded, and cross-domain conditions. In terms of efficiency, end-to-end latency, frame rate, peak video memory consumption, and edge-side power usage should all be reported, since many methods that achieve strong offline accuracy may remain impractical under real-time deployment constraints. For neural mapping and dense mapping approaches, map quality may be considered a supplementary criterion; however, rendering quality should not be used as a surrogate for localization and tracking performance.

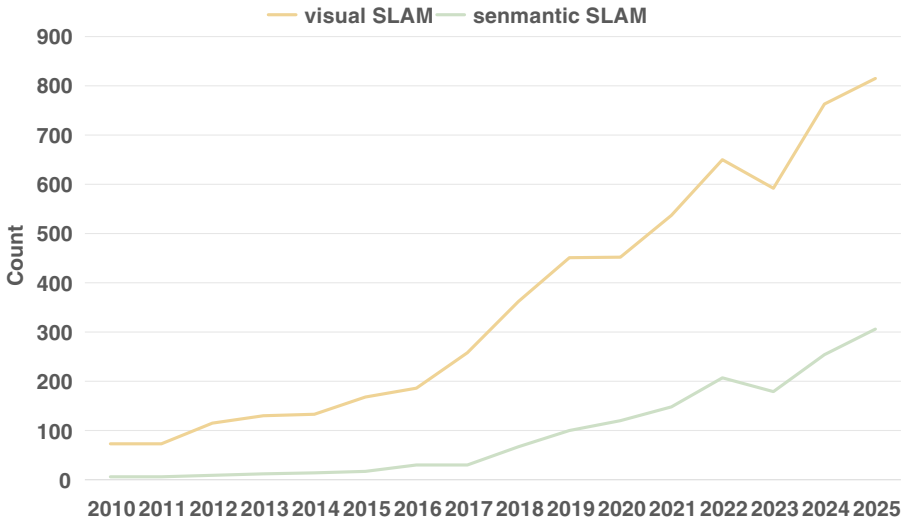

Figure 3: Publications on VSLAM and semantic SLAM in recent years.

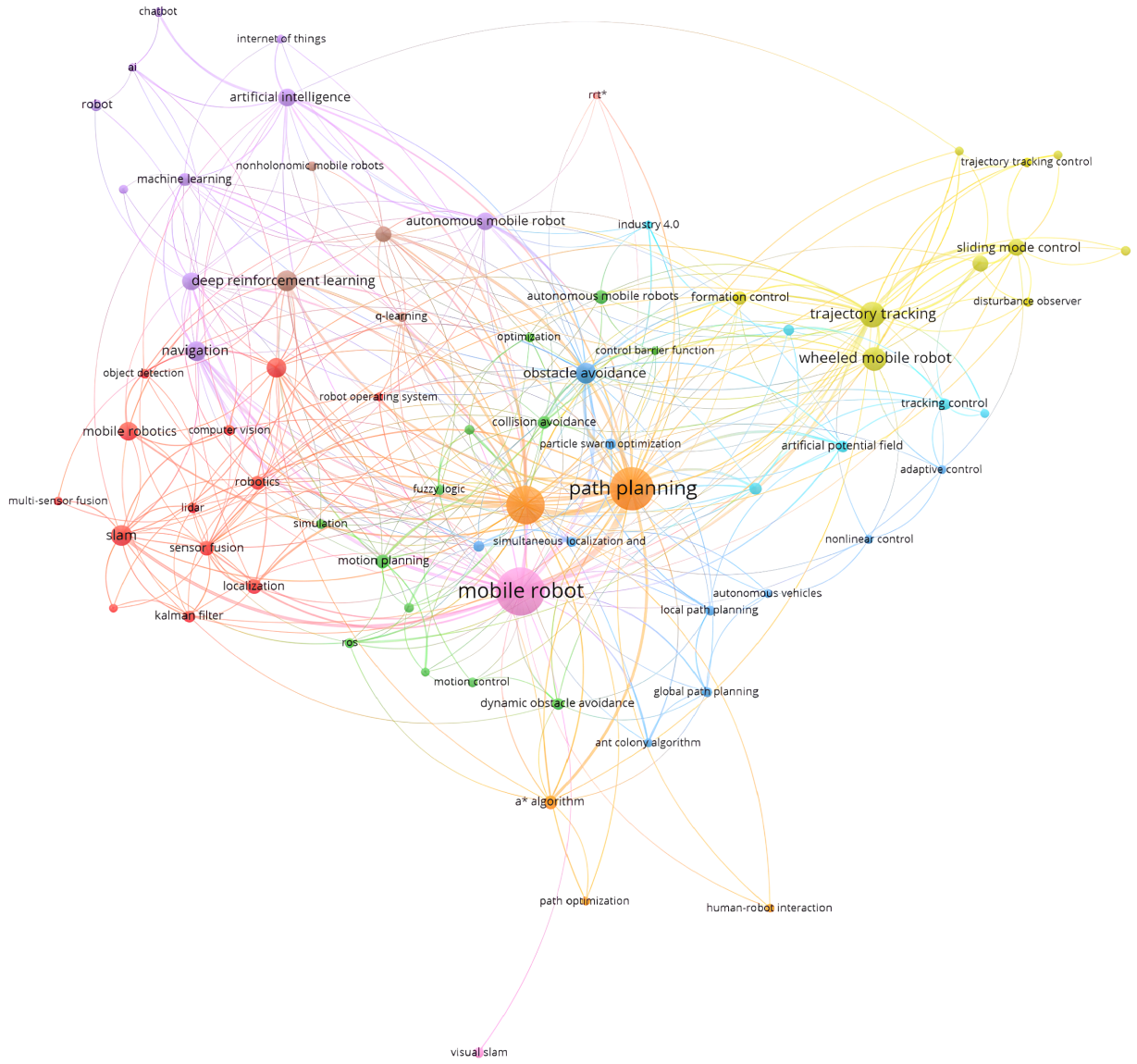

Figure 4: VSLAM keyword clustering analysis diagram. The "*" is part of the algorithm name rather than a marker of statistical significance. Specifically, RRT denotes the asymptotically optimal variant of Rapidly-exploring Random Tree, while A* algorithm refers to the classical heuristic search-based path planning method.

Furthermore, to ensure that conclusions across methods are genuinely comparable, system configurations must be explicitly specified, including input resolution, camera intrinsics, use of IMU or depth information, keyframe strategy, loop-closure activation, and optimization window size. Additional implementation details, such as the number of model parameters, the inference platform, whether the system operates on an on-demand basis, and whether in-domain fine-tuning is performed during testing, should likewise be clearly reported. Without such information, even similar metric values cannot reveal whether observed performance differences stem from the algorithmic design itself or from variations in system settings and evaluation protocols. Therefore, comparisons of deep learning-enhanced VSLAM should not be regarded as purely algorithmic comparisons but rather as joint assessments of algorithm design, system

configuration, and evaluation methodology. The discussion of representative methods in Chapter 3 is conducted under this unified baseline.

3 Deep Learning in Visual SLAM

This chapter focuses on the central questions of deep learning-enhanced VSLAM: what learning models produce, where their outputs are integrated into the geometric pipeline, and the implications and trade-offs this integration entails for accuracy, robustness, and efficiency. With fusion interfaces as the organizing principle, the discussion is structured into five categories: observation-level interfaces, constraint/prior/weight-level interfaces, solver-level interfaces, representation-level interfaces, and system-level integration interfaces. Each subsection systematically examines the limitations of traditional geometric baselines, the output forms of learning-based modules, how they are coupled with geometric constraints, and the common failure modes that may arise in practice.

3.1 Fusion-Interface Taxonomy and Boundary Rules

When deep learning is integrated into VSLAM, the most effective strategy is typically not to replace geometric solvers entirely, but to enhance the conventional "observation-constraint-optimization" chain by providing more reliable inputs and more stable optimization behavior. Based on the existing literature, learning-enhanced VSLAM can be broadly categorized into the following types of fusion interfaces.

(1) **Observation-level interface:** Representative methods, such as SuperPoint [18], SuperGlue [19], and LoFTR [20], learn key points, descriptors, and matching relationships. Related approaches further estimate dense depth and optical flow to strengthen the observability of scale and structure, or predict semantic segmentation and instance masks to suppress dynamic objects and improve loop-closure discriminability. These methods usually retain geometric solvers such as PnP, bundle adjustment (BA), and pose graph optimization (PGO). Their main advantages lie in strong interpretability and convenient integration with existing systems; however, their limitations often manifest as difficulty with cross-domain generalization and considerable real-time computational overhead.

(2) **Constraint/prior/weight-level interface:** In this class of methods, the network predicts residual confidence, uncertainty, or robust kernel parameters, which are then used to reweight matches, pixel-wise residuals, or loop closure candidates, thereby modifying the optimization convergence basin and suppressing outliers. In addition, motion or structural priors can be learned to regularize in constrained scenarios [1,4,21]. The effectiveness of such interfaces depends critically on whether the learned quantities align with geometric residuals in both scale and statistical meaning; otherwise, systematic bias may be introduced.

(3) **Solver-level interface:** This category embeds PnP, BA, PGO, or their approximations into neural networks in a differentiable manner, so that end-to-end training can jointly adapt both feature representations and the solving process to task-level objectives, such as tracking success rate and reprojection residual [1,4,22]. The main advantage is that the learning objective becomes more closely coupled to the closed-loop behavior of SLAM. Nevertheless, such methods remain challenged by numerical stability, gradient propagation, and substantial training cost.

(4) **Representation-level interface:** These methods extend conventional sparse-point or keyframe-based maps toward neural implicit representations, such as Neural Radiance Fields (NeRF) and signed distance functions (SDF) [23,24], as well as 3D Gaussian Splatting [25]. Such representations can support both high-quality rendering and dense geometric reconstruction; however, they usually incur higher computational and storage overhead, thereby imposing stricter requirements on real-time tracking and overall system design.

(5) **System-level integration interface:** This category focuses on complete VSLAM architectures in which multiple learning-enhanced modules are coordinated across tracking, mapping, relocalization, and loop closure. Representative systems, such as DROID-SLAM [22], NICE-SLAM [26], NeRF-SLAM [27], SplaTAM [28], GS-SLAM [29], VIGS-SLAM [30], and WildGS-SLAM [31], illustrate how learned observations, differentiable updates, neural maps, and system-level scheduling can be integrated into complete SLAM pipelines. The key issue in this interface is the coordination among different modules, including tracking–mapping coupling, relocalization triggering, loop-closure verification, map update scheduling, and resource management.

It should be noted that the proposed taxonomy is not intended to imply that each method belongs exclusively to a single technical family. Many recent learning-enhanced VSLAM systems combine multiple mechanisms. For example, DROID-SLAM [22] contains both learned dense correspondence estimation and differentiable bundle adjustment, while NICE-SLAM [26], NeRF-SLAM [27], SplaTAM [28], and GS-SLAM [29] combine neural map representations with integrated tracking and mapping pipelines. To avoid ambiguity, this review assigns each method to a primary category according to its dominant fusion interface, namely the point at which the learning output most directly enters or modifies the geometric SLAM pipeline. Secondary functions are discussed as cross-category attributes rather than as the basis for repeated classification.

Specifically, as shown in Fig. 5, a amethod is classified as an observation-level interface if the network output is mainly used as features, matches, depth, optical flow, or semantic masks for subsequent geometric estimation. It is classified as a constraint/prior/weight-level interface if the network mainly provides confidence, uncertainty, residual weights, robust kernels, or motion and structural priors to regularize residual optimization. It is classified as a solver-level interface if geometric solvers such as PnP, bundle adjustment, pose graph optimization, dense BA, or iterative update modules are embedded into a differentiable or learnable computational graph. It is classified as a representation-level interface if the main contribution lies in replacing or extending the map with neural implicit fields, signed distance functions, NeRF-based maps, or 3D Gaussian primitives. Finally, it is classified as a system-level integration interface if multiple learning-enhanced modules are jointly trained or tightly coupled to coordinate tracking, mapping, relocalization, or loop closure within a complete SLAM system.

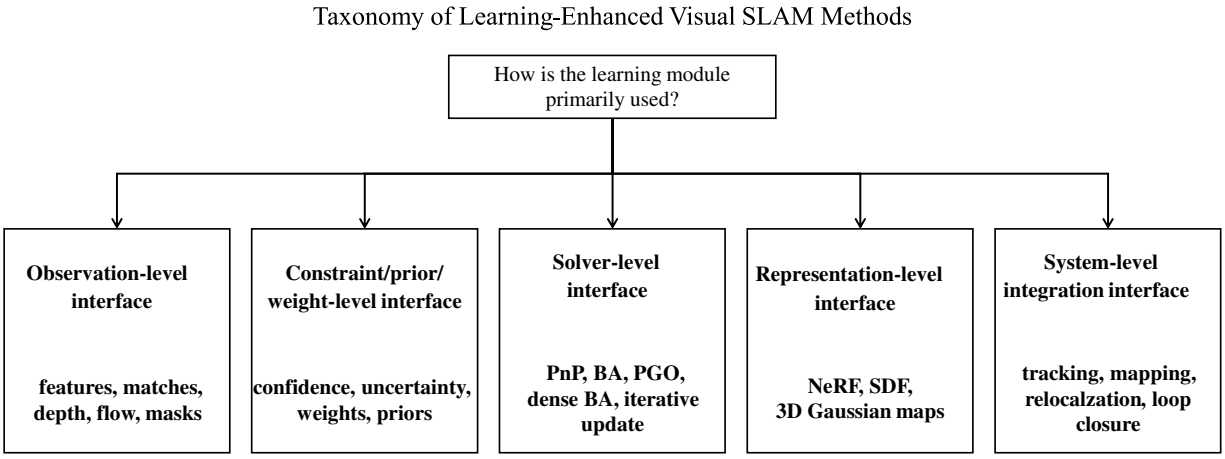

Figure 5: Decision tree for assigning learning-enhanced VSLAM methods to their primary fusion-interface categories.

3.2 Observation-Level Interface

This section focuses on learning-based local features and matching, which constitute one of the most readily deployable enhancement pathways in VSLAM and correspond to the core component of the Observation-Level Interface in Table 3. The discussion is organized around three key stages: feature representation learning, learning-based matching with geometric-consistency filtering, and loop-closure detection with visual relocalization. Particular attention is given to transforming the outputs of deep models into observations or weighting cues suitable for geometric solvers such as PnP, bundle adjustment (BA), and pose graph optimization (PGO). The corresponding performance gains, practical advantages, and common failure modes are also summarized.

Table 3: Classification and trade-offs in the integration of deep learning and VSLAM.

Fusion Interface	Typical Learning Outputs	Insertion Point in the Geometric Pipeline	Primary Advantages	Primary Risks/Costs
Observation-level interface	Key points, descriptors, and matches; depth and optical flow; semantic masks	Front-end matching, relocalization, and loop closure; observations for PnP/BA	Improved robustness; strong interpretability; easy integration with existing systems	Cross-domain generalization issues; real-time computational overhead; bias from training data
Constraint/prior/weight-level interface	Confidence and uncertainty estimates; robust kernel parameters; motion and structural priors	Residual reweighting, outlier rejection, and optimization regularization	Enlarged convergence basin; suppression of dynamic disturbances and noise	Bias caused by scale or statistical misalignment
Solver-level interface	Differentiable PnP/BA/PGO or learnable iterative update modules	Inside the optimizer, where the solving process becomes learnable	Better consistency between learning objectives and closed-loop SLAM behavior; joint adaptation	Numerical instability; high training cost
Representation-level interface	Dense representations such as NeRF, SDF, and 3D Gaussian Splatting (3DGS)	Mapping and rendering modules	High-quality reconstruction and visualization; convenient semantic fusion	High memory and computational cost; difficulty in real-time tracking
System-level integration interface	Coordinated tracking, mapping, relocalization, and loop-closure modules	Complete VSLAM pipeline and system scheduling	Cross-module coordination; improved long-term usability; deployment-aware integration	Pipeline complexity; debugging difficulty; resource contention; failure propagation

Conventional VSLAM front ends rely predominantly on hand-crafted features, such as ORB, SIFT, and SURF, together with descriptor-based nearest-neighbor matching, followed by RANSAC-based geometric verification. Under challenging conditions, including weak textures, repetitive patterns, severe illumination changes, and motion blur, as illustrated in Fig. 6, the repeatability and discriminative capability of such features deteriorate substantially. This degradation often leads to a rapid increase in outliers, tracking failures, and false loop-closure detections [2,6,32].

Learning-based local features utilize data-driven learning to identify more stable key points and descriptors. SuperGlue [19] was among the first methods to combine graph neural networks with optimal transport for global contextual reasoning on the keypoint graph. More importantly, it produced high-quality matching confidence scores that can be directly incorporated as observation weights in bundle adjustment (BA) and pose graph optimization (PGO), thereby significantly improving the inlier ratio in

scenarios with large viewpoint changes and weak textures. With SuperPoint features, it reports outdoor pose AUC@5°/10°/20° of 34.18/50.32/64.16 and matching precision of 84.9%, compared with 9.80/18.99/30.88 and 22.5% for mutual nearest-neighbor matching. However, its Transformer-style message passing introduces considerable computational overhead, which can easily become a latency bottleneck in real-time VSLAM. The corresponding processing pipeline is illustrated in Fig. 7.

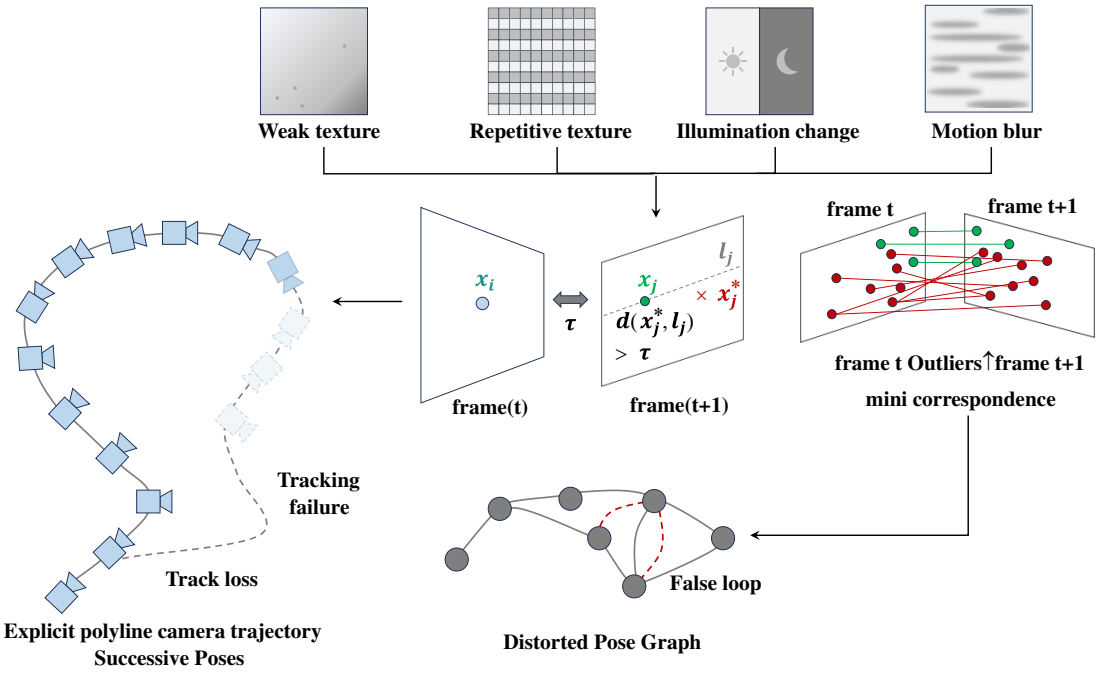

Figure 6: Failure mechanism of a traditional VSLAM front-end under degraded conditions: feature ambiguity induces structured mismatches that violate peripolar consistency, leading to outlier explosion, tracking loss, and false loop closure.

To alleviate this limitation, LoFTR [20] proposed a Transformer-based coarse-to-fine dense-matching framework that operates without explicit keypoint detection, thereby eliminating reliance on conventional detectors. This design yields clear advantages over SuperGlue in low-texture and repetitive-texture regions, where it provides more reliable structural observability. Nevertheless, the computationally intensive nature of LoFTR results in higher inference latency and GPU memory consumption, which restricts its deployment on resource-constrained platforms. Subsequent methods, including MatchFormer [33] and AspanFormer [34], introduced targeted improvements to address the efficiency limitations of LoFTR. The lite MatchFormer reports only 45% GFLOPs, a +1.3% precision gain, and a 41% running-speed boost over the previous best indoor pose-estimation method, while ASpanFormer adaptively adjusts the attention span to retain long-range matching accuracy with lower redundant computation. Together, these methods have advanced detector-rematching toward real-time applicability.

At the feature representation level, ALIKED [35] streamlines keypoint and descriptor extraction by introducing a sparse deformable descriptor head that learns support feature locations for sparse key points, along with a relaxed neural reprojection loss to improve training efficiency. This design improves repeatability with respect to changes in viewpoint and illumination, although systematic bias may still persist under extreme variations in appearance. To address the real-time limitations of the SuperGlue [19] and LoFTR [20] families, LightGlue [36] revisits the design of SuperGlue by introducing adaptive depth and width pruning,

as well as difficulty-aware early stopping. Its official benchmark reports 150 FPS at 1024 key points and 50 FPS at 4096 key points on an RTX 3080, corresponding to about 4–10× speedup over SuperGlue while maintaining competitive matching accuracy. Its confidence outputs can be incorporated into geometric weighting or outlier rejection, although severe seasonal and day-night appearance changes still require subsequent geometric verification. From the perspective of global descriptors, EigenPlaces [37] complements this line of research by improving cross-view VPR while requiring 60% less GPU memory for training and using 50% smaller descriptors than previous strong baselines, thereby providing more reliable initial candidates for subsequent loop closure modules.

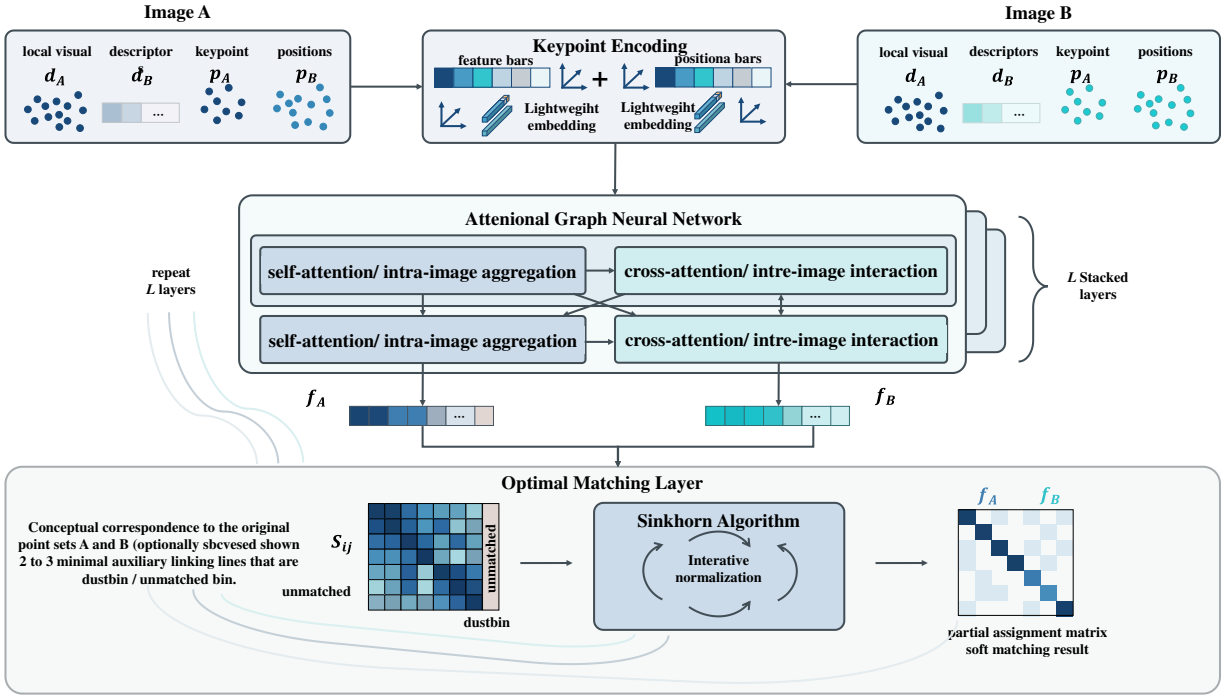

Figure 7: Basic framework of SuperGlue. Local descriptors and key point positions from two images are first embedded by a key point encoding module and then processed by an attentional graph neural network. The network alternates between self-attention for intra-image context aggregation and cross-attention for inter-image feature interaction. An optimal matching layer subsequently computes pairwise matching scores and applies Sinkhorn normalization to obtain a partial soft assignment matrix, enabling robust keypoint matching with explicit handling of unmatched points.

In 2024, RoMa [38] addressed structural outliers induced by severe illumination changes, seasonal variations, and dynamic scene perturbations by proposing a robust dense-matching framework built on frozen DINOv2 for coarse feature extraction, a ConvNet for refinement, and a Transformer decoder. By combining regression-by-classification with a robust loss formulation, it achieved a 36% improvement on the WxBS benchmark. The predicted anchor probabilities can be directly used as residual weights in BA; however, this method relies heavily on large-scale pretraining and requires substantial computational resources for on-device deployment. In the same year, Efficient LoFTR [39] specifically targeted LoFTR's latency bottleneck. By employing aggregated attention and adaptive token selection, it enables semi-dense matching at a speed comparable to sparse matching. In low-texture scenarios, it achieves an inlier rate comparable to that of LightGlue while delivering a 2.5-fold speedup, thereby offering a practical compromise for highly real-time platforms such as onboard systems and aerial robots. The corresponding workflow is illustrated in Fig. 8.

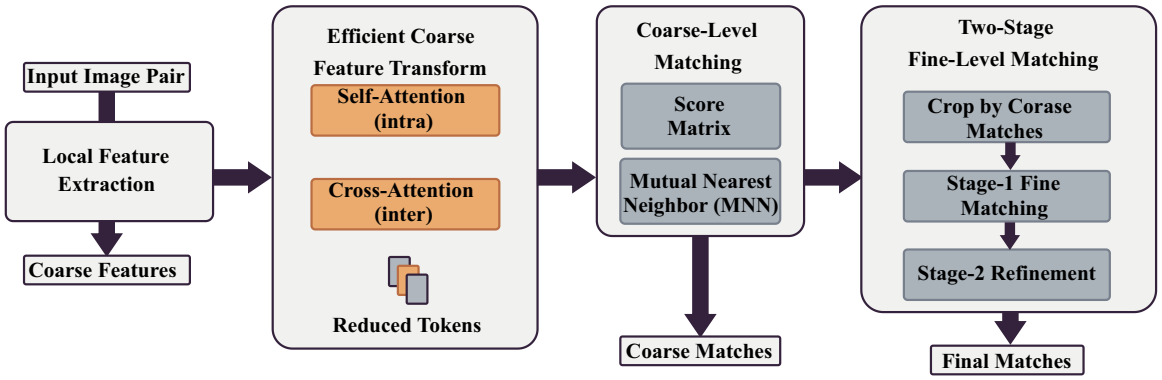

Figure 8: Core framework of efficient LoFTR: local features are extracted from the input image pair, transformed through efficient intra-image self-attention and inter-image cross-attention with reduced tokens, matched at the coarse level using a score matrix and mutual nearest-neighbor selection, and finally refined by a two-stage fine-level matching process to obtain accurate final correspondences.

Learning-based matching can substantially suppress structural outliers during geometric consistency filtering. At the stage of learning-based matching and geometric verification, conventional approaches, such as nearest-neighbor matching combined with ratio tests and RANSAC, are prone to producing structural outliers in scenes with large viewpoint changes, repetitive textures, low illumination, or heavy dynamic occlusion. The progression from SuperGlue [19] to Efficient LoFTR [39] and RoMa [38] has significantly improved the inlier rate and reduced the number of outliers by precisely modeling global context and predicting confidence or probability scores.

From the perspective of fusion mechanisms, the essential role of learning-based features is not to replace geometric constraints, but to provide higher-quality observations for subsequent estimation. A common strategy is to feed the outputs of these learning-based methods into PnP or bundle adjustment (BA), including the confidence scores produced by SuperGlue and LightGlue, the dense correspondences generated by LoFTR and Efficient LoFTR, and the probability maps predicted by RoMa, while further mapping such confidence or probability estimates to observation weights or employing them for outlier rejection. Another strategy relies on global descriptors, as exemplified by Patch-NetVLAD [40], TransVPR [41], and SALAD [42], which perform learning-based retrieval during loop closure and relocalization, followed by local geometric verification to complete the loop, thereby balancing recall and precision.

Nevertheless, learning-based feature representations still face challenges in cross-domain generalization, real-time performance, and statistical alignment with back-end optimization. When the training data are not well aligned with the actual camera intrinsics, noise characteristics, or scene distributions encountered in deployment, the learned matcher may exhibit systematic bias. In addition, dense matching approaches, such as RoMa and early versions of LoFTR, often impose substantial computational overhead when deployed on devices alongside large models. Furthermore, if the predicted confidence scores or weights are statistically inconsistent with geometric residuals, the optimization process may become biased or even diverge.

A robust engineering strategy is hierarchical deployment. During routine tracking, lightweight key points are retained to ensure real-time operation; when weak textures, rapid motion, or relocalization demands arise, an efficient matching module is activated as an enhancement. The associated latency can then be controlled through keyframe feature caching, reduced input resolution, or the use of distilled and quantized models. From the perspective of evaluation, it is advisable to report not only matching-level metrics, such as inlier ratio and reprojection error, but also system-level measures, including tracking success

rate, relocalization success rate, absolute trajectory error (ATE), relative pose error (RPE), and the false positive rate in loop closure tests. Only under such a comprehensive evaluation protocol can one determine whether front-end enhancement has truly translated into overall gains in SLAM.

Loop closure detection plays a decisive role in maintaining global consistency during long-term operation and in large-scale environments. Traditional bag-of-words (BoW) methods, including DBoW2, are highly sensitive to appearance variation. In contrast, learning-based visual loop closure enhances retrieval robustness to changing appearances by learning global descriptors and typically employs a more reliable two-stage integration strategy comprising retrieval and verification. Specifically, learning-based global descriptors are first used to retrieve a set of candidate keyframes to improve recall; local geometric verification is then performed to confirm true loop closures and generate constraints for pose graph optimization (PGO), thereby controlling the false-positive rate. The central idea of this paradigm is that the learning module proposes candidates, while geometric consistency makes the final decision.

Early methods often produce false positives in scenes with repetitive textures because residual aggregation alone may provide insufficient global discriminability. To address this issue, CosPlace [43] reformulates visual place recognition as a scalable classification problem by partitioning geo-referenced images into spatial classes and training descriptors with a cosine-margin objective, rather than relying on costly pairwise or triplet mining. This encourages nearby places to form compact descriptors while separating different spatial regions in the normalized embedding space, achieving a Recall@1 of 79.6% on the MSLS benchmark. Nevertheless, CosPlace remains a global retrieval method and cannot directly guarantee geometrically valid loop-closure constraints; visually similar but spatially distinct places may still be retrieved under repetitive structures, viewpoint changes, or dynamic occlusions.

To address burst-aware effects and sequential noise, VLAD-BuFF [44] proposes a fast, burst-aware feature aggregation method that applies adaptive weighting to abrupt variations in image sequences, thereby improving robustness to seasonal changes and dynamic disturbances. It reports state-of-the-art recall on nine public datasets while maintaining high recall with 12× reduced local feature dimensions. DDA-VPR [45] further extends this idea to dual-domain aggregation in both spatial and frequency spaces. By exploiting frequency-domain cues to capture global structural information and bridging domain gaps through triple fusion, it produces a more discriminative global representation and outperforms SALAD on challenging benchmarks. Pair-VPR [46], in turn, combines place-aware masked-image pretraining with contrastive pair classification to enhance cross-domain generalization and robustness under dynamic interference, reporting state-of-the-art VPR performance across five benchmark datasets.

More recent studies [47–54] have increasingly incorporated semantic and dynamic information into the loop closure pipeline, for example by filtering dynamic regions during retrieval or matching, or by relying on static structural semantics as a more stable basis for loop closure. Table 4 provides a brief summary of these representative methods. It should be emphasized that, in scenarios involving large viewpoint changes, a high proportion of dynamic content, or weak textures, candidate keyframes may fail to pass geometric verification. Accordingly, successful visual place recognition does not necessarily imply the availability of valid geometric constraints. For this reason, the loop closure module is better deployed alongside learning-based matchers within a multi-stage gating mechanism at the system level. From an evaluation perspective, it is more meaningful in practical settings to jointly assess loop closure recall, false positive rate, and the number of valid loop closures that pass geometric verification and successfully trigger PGO. In addition, these statistics should be reported separately on cross-temporal datasets and dynamic-interference datasets, so that the effects of appearance variation and dynamic noise can be more clearly distinguished.

Table 4: Representative learning-based loop closure and visual place recognition methods.

Method	Year	Contribution
SVS-VPR [47]	2024	Integrate semantic filtering and spatial constraints to improve VPR recall and precision in challenging dynamic environments.
Semantic Boundary Constrained Network [48]	2025	Introduce semantic boundary cues to suppress dynamic regions and support loop detection under large viewpoint changes.
SRD2-VPR [49]	2025	Enforce semantic feature aggregation and query rejection to improve verification success in dynamic environments.
ADTA-VPR [50]	2025	Use deformable token aggregation and semantic masks to strengthen static-structure cues for loop closure.
SCM-PR [51]	2025	Combine semantic filtering and multi-stage geometric verification for more reliable cross-modal loop closure.
High-level adaptive feature enhancement VPR [52]	2026	Use adaptive feature enhancement and attention masks to improve retrieval robustness across temporal variations.
Ensemble-Based Event Camera VPR [53]	2026	Combine event-sequence matching with semantic filtering to improve loop closure under varying lighting and dynamic conditions.
Context-Based Visual-Language VPR [54]	2025/2026	Use vision-language context and geometric verification to separate appearance variation from motion interference.

3.3 Constraint/Prior/Weight-Level Interface

This section discusses learning-enhanced VSLAM methods that condition the geometric optimization process through priors, residual weights, uncertainty estimates, and dynamic-scene masks. In this interface, the learning module modifies how residuals are initialized, selected, weighted, or regularized. Typical outputs include relative-motion priors, depth- or flow-based initialization cues, semantic masks, instance-level dynamic-object masks, pixel-level uncertainty, match-level confidence, and structural priors. These quantities are usually injected into PnP, bundle adjustment (BA), pose graph optimization (PGO), or photometric alignment as auxiliary constraints or adaptive weights. Their main function is to enlarge the convergence basin of geometric solvers, suppress unreliable residuals, and reduce the propagation of dynamic or uncertain observations into trajectory drift. The effectiveness of this interface depends strongly on the consistency between learned quantities and geometric residuals. Overconfident weights may amplify biased observations, while overly conservative weights may weaken the contribution of learning modules.

In underconstrained scenarios, such as monocular scale ambiguity, low-parallax motion, weak texture, and rapid camera movement, learned depth, optical flow, and motion priors can provide useful initialization or regularization cues [2,6]. Monocular depth and optical flow reduce the search space for correspondence and photometric alignment, while learned relative-pose or velocity priors guide local pose estimation prior to nonlinear refinement. In dynamic scenes, semantic or instance masks further exclude or downweight regions that violate the static-world assumption, improving the robustness of residual construction.

Motion priors and initialization cues. Within the constraint/prior/weight-level interface, learning-based motion priors are a common mechanism for improving pose initialization and regularization during optimization. TartanVO [55] proposed an end-to-end learning-based visual odometry system that employs a CNN-LSTM recurrent architecture to predict relative pose and velocity for initialization. This design substantially enlarges the convergence basin of PnP and bundle adjustment (BA), thereby reducing tracking

interruptions in fast-motion scenarios. However, because it relies heavily on motion patterns represented in the training data, it is prone to systematic errors in situations involving sharp drone maneuvers or handheld camera shake.

To achieve uncertainty adaptation in dynamic environments, UP-SLAM [56] proposed an adaptive structured Gaussian SLAM framework in 2025. Through a parallelized architecture that decouples tracking from mapping and predicts pixel-level uncertainty, it dynamically adjusts overconfident or overly conservative weights within geometric residuals, thereby effectively suppressing error propagation under dynamic occlusion. UP-SLAM reports a 59.8% improvement in localization accuracy and a 4.57 dB gain in PSNR while maintaining real-time RGB-D SLAM performance in dynamic environments. The corresponding workflow is illustrated in Fig. 9.

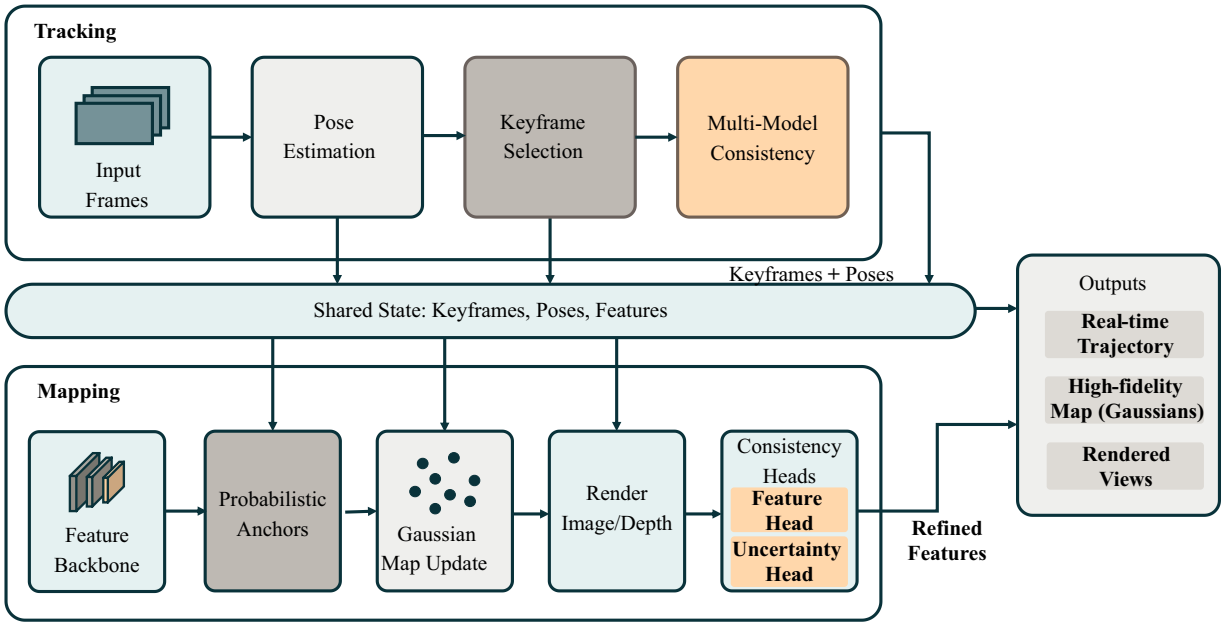

Figure 9: Uncertainty-aware tracking and mapping workflow in UP-SLAM. The framework couples a tracking pipeline and a Gaussian mapping pipeline through a shared state of keyframes, poses, and features. Pose estimation and keyframe selection provide geometric constraints for online tracking, while probabilistic anchors and Gaussian map updates build a high-fidelity scene representation. Rendered image/depth predictions are assessed by feature and uncertainty consistency heads, allowing the system to suppress unreliable observations and improve robustness in trajectory estimation, map reconstruction, and novel-view rendering.

At the same time, DMS-SLAM [57] focuses on mechanisms to handle dynamic interference by generating real-time semantic masks with a depth-masking segmentation network, thereby excluding or downweighting dynamic regions. When integrated with conventional bag-of-words (BoW) frameworks, this design achieves more than a 51% efficiency improvement and further alleviates the residual limitations observed in UP-SLAM [56] in scenarios with extremely high dynamic content. Nevertheless, in weakly textured scenes and under severe illumination variation, it still requires integration with optical flow priors. ADEmono-SLAM [58], by contrast, specialized in absolute depth estimation in 2025. By directly predicting absolute scale and combining it with relative-depth alignment, this approach mitigates cross-domain scale drift in large-scale outdoor environments and provides monocular systems with a more explicit, hard-constrained initialization.

With respect to optical flow and motion priors, a 2025 deep learning framework [59] further employed a CNN-based optical flow network to directly construct SLAM photometric residuals and initialize matching, thereby reducing the search space under rapid motion and enlarging the convergence basin. However, its sensitivity to data distribution still requires geometric gating for effective control. In the same year, DI-SLAM [60] addressed dynamic indoor environments by integrating RGB-D semantics with geometric constraint pruning, thereby achieving more stable visual-inertial odometry (VIO) optimization and reducing the sensitivity to motion priors.

Learning outputs can be safely incorporated into geometric optimization only when they are both scale-consistent and statistically meaningful [4]. A common strategy is to map the predicted pixel-level or match-level uncertainty, denoted by σ, to a residual weight w, and then construct a weighted residual under a robust kernel $\rho(\cdot)$:

$$\min \sum_i \rho(\omega_i ||r_i||^2)$$

However, network-predicted uncertainty does not necessarily correspond to the true geometric noise scale, and a high confidence score should not be interpreted directly as a calibrated probability of geometric correctness. If σ is excessively small, the model becomes overconfident, which may cause wrong matches, biased depth estimates, or dynamic-object residuals to dominate BA/PGO and induce drift. Conversely, if σ is excessively large, the learning module's contribution approaches zero, making it difficult to achieve system-level benefits [1,4]. This problem becomes more severe when heteroscedastic uncertainty is misspecified, especially when the training distribution differs from the deployment environment.

During online operation, residual feedback can be used for adaptive recalibration [26,56]. However, uncertainty outputs should not be directly treated as optimization weights unless they have been calibrated and verified. Practical calibration strategies include temperature scaling or variance rescaling on validation sequences, variance clipping to avoid unbounded weights, negative-log-likelihood calibration, and reliability analysis using reliability diagrams or expected calibration error (ECE). After calibration, geometry-aware consistency checks should still be retained, such as inlier-count thresholds, reprojection-error bounds, Mahalanobis residual consistency tests, χ^2 gating, and scale-consistency tests, to prevent miscalibrated learning outputs from entering global optimization. When these checks fail, the system should fall back to RANSAC-based rejection or uniform weighting. In this way, uncertainty prediction is converted from a purely learned confidence score into a calibrated and geometry-compatible interface that can work more reliably with reprojection residuals and robust kernels.

Taken together, the strong performance of systems based on learned depth, optical flow, and motion priors often depends on the statistical alignment between learning outputs and geometric residuals. Consequently, in practical deployment, uncertainty calibration must be combined with geometric constraints to prevent biases in learned observations from amplifying into trajectory drift along the optimization chain. From the perspective of real-time execution, pose estimation is typically a critical decision-making module, and therefore the model architecture and scale must be carefully controlled. Current practical strategies include combining lightweight encoders with multi-task heads, activating heavier models only on demand during keyframe processing or failure recovery, and employing quantization, pruning, and hardware acceleration to prevent uncontrolled latency on edge devices. Accordingly, when evaluating such models, in addition to key metrics such as absolute trajectory error (ATE) and relative pose error (RPE), one should also consider the number of tracking interruptions under dynamic and varying illumination conditions, relocalization time, and robustness curves across different motion speeds or disparity levels, so as to more comprehensively assess the usability gains brought by learning-enhanced models.

Dynamic masks, semantic prior, and object-level constraints. Dynamic and semantic information forms an important extension of the constraint/prior/weight-level interface. In dynamic scenes, learning modules are used to identify unreliable regions, provide semantic priors, and construct object-level constraints. These outputs influence the geometric pipeline by modifying residual selection, residual weighting, map update rules, and loop-closure verification. Dynamic scenes and semantic information are therefore critical factors that determine the practical usability of VSLAM systems in real-world environments.

As illustrated in Fig. 10, the presence of dynamic objects can severely degrade the reliability of visual observations and prevent the camera from capturing stable geometric information. Consequently, mitigating the adverse impact of dynamic objects on SLAM systems has become a major objective in recent research. In this context, the value of deep learning does not lie in replacing geometric optimizers such as pose graph optimization (PGO) or bundle adjustment (BA), but rather in providing interpretable dynamic-semantic cues that enable the system to identify which observations should be downweighted or rejected and which scene structures remain sufficiently stable for reliable estimation.

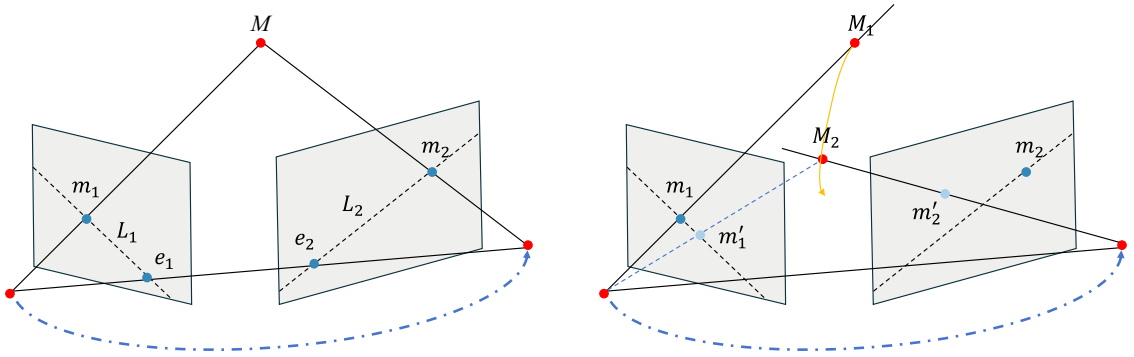

Figure 10: Traditional methods determine whether an object is in motion by exploiting geometric constraints. As illustrated in the left panel, the point remains stationary in space, and the corresponding spatial transformation can therefore be estimated accurately. In contrast, as shown in the right panel, the motion of the spatial point from to introduces systematic error into the geometric estimation process.

In dynamic scenes, constraint/prior/weight-level methods have gradually evolved from hard exclusion of moving objects toward soft residual weighting and semantic regularization. DynaSLAM II [61] tightly couples multi-object tracking with SLAM by combining instance segmentation, ORB features, and joint bundle adjustment for both camera and object trajectories. On the KITTI tracking dataset, it reduces the mean ATE from 2.33 m with ORB-SLAM2 to 1.54 m, while keeping the mean translational RPE at 0.053 m/frame and rotational RPE at 0.043°/frame. On the KITTI raw dataset, it reports a mean ATE of 1.44 m, compared with 3.73 m for ORB-SLAM2 and 1.30 m for DynaSLAM, showing that tracking dynamic objects can improve ego-motion estimation when dynamic agents provide useful geometric constraints. Empty Cities [62] takes a different approach by synthesizing dynamic-object-free urban images using a GAN and evaluating the effects on visual odometry, place recognition, and multi-view stereos. Its reported experiments show that hallucinated static images can improve localization and mapping consistency in dynamic urban scenes, but this benefit comes at the cost of additional image reconstruction and potential inpainting artifacts.

In the same year, DOE-SLAM [63] addressed the reconstruction overhead of Empty Cities by introducing a soft-weighting mechanism with dual validation based on semantic and motion consistency. This mechanism adjusts observation weights according to motion consistency, thereby retaining more measurements while avoiding excessive pruning. To further balance the trade-off between hard pruning and

soft weighting, Blitz-SLAM [64] proposed a two-stage semantic-geometric validation framework in 2022. Specifically, hard pruning is adopted when the proportion of moving objects is high, whereas soft weighting is used when the dynamic ratio is relatively low, leading to a notable improvement in real-time performance. Nevertheless, the method still suffers from semantic missegmentation and cannot fully avoid false pruning.

To align optimization objectives more closely with practical tasks, RGB-D SLAM with Multilevel Semantic Information [65] introduced multilevel semantic constraints that integrate object detection, optical flow, inter-object geometric relationships, and 3D point cloud clustering. This method is evaluated by both trajectory estimation and object-level dense semantic mapping, showing that semantic constraints can supplement point-feature residuals in dynamic RGB-D scenes. In 2023, Semantic Visual SLAM Using Deep Learning for Dynamic Scenes [66] employed an improved Mask R-CNN with geometric verification to mitigate over-pruning due to semantic missegmentation. On TUM dynamic sequences, it reduces the average ATE RMSE by 96.39% and ATE standard deviation by 95.03% compared with ORB-SLAM2 in highly dynamic environments. It also reduces relative displacement RMSE by 96.39% and relative rotation RMSE by 90.07%, while maintaining an average tracking time of about 60.73 ms per frame.

Recent semantic SLAM systems also extend semantic weighting and regularization mechanisms to neural or Gaussian maps. SNI-SLAM [67] constructs updatable dynamic semantic maps using NeRF-like implicit networks and is evaluated on Replica and ScanNet, achieving state-of-the-art performance in tracking, mapping, and semantic segmentation compared with recent NeRF-based SLAM methods. GS4 [68] further integrated 3D Gaussian Splatting with semantic fusion and proposed a generalizable sparse Gaussian-Splatting-based semantic SLAM framework. By introducing a renderable layer and a low-frequency update thread into the dynamic semantic map, it substantially alleviated the resource competition between mapping and localization. Compared with prior approaches, it runs about 10 times faster, uses about 10 times fewer Gaussians, and achieves state-of-the-art performance across color rendering, depth reconstruction, semantic mapping, and camera tracking. In 2025, object-level semantic modeling has been explored to improve long-term consistency in dynamic and weak-texture environments. YOSO-SLAM [69] addressed the challenge of object-level scale alignment by proposing a YOLO-based Semantic Object SLAM framework, which supports object-level 3D semantic maps and pose priors, thereby improving long-term consistency under weak-texture conditions.

Semantic structure-based constraints. Semantic information can be used not only to filter moving objects but also to construct higher-level, more stable geometric constraints. For example, semantic structures such as planes, object boundaries, and road markings exhibit high stability in both indoor and outdoor scenarios. These elements can be introduced into graph optimization as long-term constraints. At the same time, the back-end supplements point-feature residuals with planar consistency, object-scale and pose priors, and semantic-boundary constraints, thereby enhancing stability in environments with weak or repetitive textures. As early as 2005, the concept of semantic maps had already been introduced in the literature [70]. As shown in Fig. 11, this map consists of two parallel layers: a spatial layer and a semantic layer. This representation endows robots with a form of environment-level reasoning capability analogous to that of humans; for example, a bedroom can be inferred as a room containing a bed.

Compared with point features, object-level constraints offer superior reusability and interpretability. However, their effectiveness is fundamentally limited by challenges in data association and scale alignment. In particular, variations in instance identities for the same object across different viewpoints, detection failures caused by occlusion, and differences in category-specific priors all affect the reliability of such constraints. As a result, in practical systems, semantic constraints are more commonly used as weak regularizers for loop-closure triggering and subgraph connectivity, rather than as a complete replacement for point-based bundle adjustment (BA).

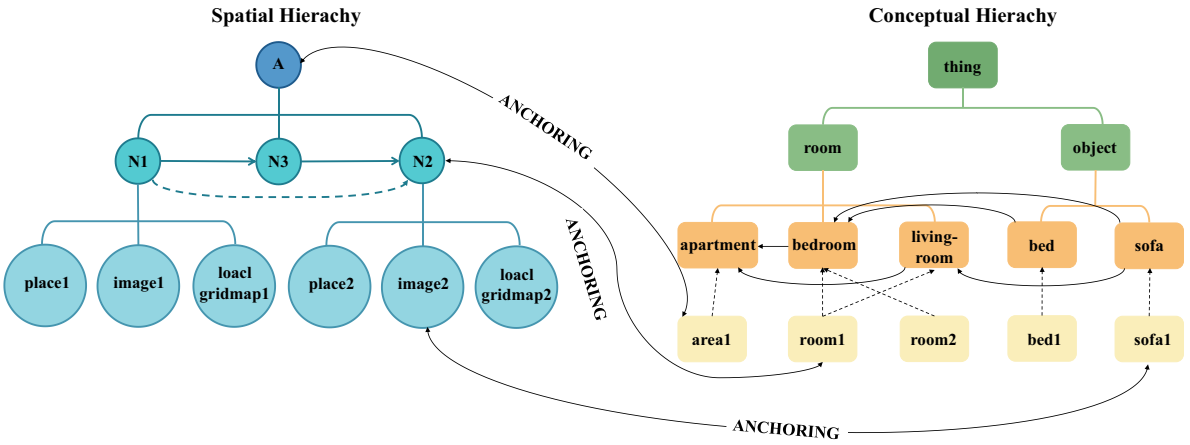

Figure 11: The semantic map concept mentioned in Galindo's article.

When a system is expected to operate over long-time horizons, the principal advantage of semantic maps lies in their ability to support an updatable world model. Such a model can distinguish static structures from variable objects, while attenuating or reconstructing the variable components during map maintenance. In practice, this is typically implemented through a hierarchical map-management strategy, in which a stable geometric skeleton is maintained at the lower layer, an object-level semantic layer is maintained at the upper layer, and consistency checks, together with incremental updates, are performed during cross-temporal revisits. Table 5 summarizes a set of representative VSLAM algorithms proposed in recent years that exploit deep neural networks, semantic constraints, and object-map integration to improve performance in dynamic environments.

Table 5: Representative dynamic and semantic constraint methods for learning-enhanced VSLAM.

Method	Year	Contribution
LVI-ObjSemantic [71]	2025	Fuse LiDAR, vision, and IMU to support object-level semantic mapping and cross-temporal updates in dynamic outdoor scenes.
SG-SLAM [72]	2025	Use object-level semantic maps as weak constraints for loop closure, submap connection, and long-term consistency maintenance.
Geometric Constraints and Semantic Optimization SLAM [73]	2025	Combine semantic optimization with geometric constraints to improve stability and association in dynamic scenes with repetitive textures.
Monocular Object-Level SLAM with Joint Semantic Segmentation [74]	2025	Combine semantic segmentation and depth estimation to build object-level priors for hierarchical mapping and temporal consistency.

However, semantic mapping still faces two systemic challenges. The first is the inconsistency of semantic evolution caused by the temporal and spatial variability of semantic models. The second is the resource competition between dynamic map updating and real-time localization. Therefore, in practical systems, mapping and localization should be decoupled at the architectural level. At the same time, semantic updates are preferably handled by low-frequency background threads or cloud-assisted collaboration so as to avoid interfering with the real-time tracking pipeline.

Beyond semantic information itself, the manner in which maps are represented has also undergone substantial transformation. The development has progressed from sparse point clouds and keyframe-based representations to dense reconstruction, and further toward renderable maps based on neural implicit representations and 3D Gaussian Splatting. These mechanisms and system costs are associated with the representation and mapping layers and are discussed in Section 3.5.

3.4 Solver-Level Interface

Solver-level interfaces are methods in which geometric state estimation is embedded in differentiable or learnable computational graphs. Typical mechanisms include differentiable PnP, differentiable bundle adjustment (BA), differentiable pose graph optimization (PGO), dense BA, and learnable iterative update modules. In these methods, learning directly affects the optimization process through pose updates, residual refinement, edge weighting, or recurrent state correction.

Early solver-level methods mainly focused on differentiating classical geometric operators. BA-Net [75] is a representative example of this direction. It introduces a differentiable bundle adjustment layer into a dense structure-from-motion framework and optimizes feature-metric residuals across multiple views. On ScanNet, BA-Net reduces the rotation error to 1.018° and the translation error to 3.39 cm, compared with 3.791° and 10.81 cm for DeMoN, 4.409° and 15.50 cm for photometric BA, and 8.560° and 21.40 cm for geometric BA. It also improves depth estimation, reducing the absolute relative depth error to 0.161, compared with 0.238 for DeMoN, 0.231 for photometric BA, and 0.268 for geometric BA. These results indicate that feature-metric BA can turn learned representations into geometry-aware optimization variables. Its limitation is that the iterative BA layer and the local optimization window increase computational cost, making direct, real-time, large-scale SLAM deployment difficult.

DeepV2D [76] further extends solver-level learning by converting structure-from-motion components into trainable modules. It alternates between camera motion estimation and depth estimation, enabling local geometric refinement within an end-to-end differentiable architecture. In later SLAM comparisons, DeepV2D is often used as a learning-based pose-estimation baseline; for example, DROID-SLAM [22] reports about 90% lower ATE than DeepV2D on TUM-RGBD, while a DPV-SLAM comparison reports EuRoC average ATE values around 1.2–1.3 m for DeepV2D variants. These results suggest that differentiable SfM improves multi-view consistency, but its iterative inference remains computationally heavy and less robust than later recurrent BA-based SLAM systems.

DROID-SLAM [22] is a representative solver-level system. It constructs dense correspondences through a RAFT-like correlation volume and couples recurrent update modules with dense bundle adjustment. Feature matching, residual construction, edge weighting, and pose refinement are therefore performed within an iterative optimization loop. Quantitatively, DROID-SLAM reports an average monocular EuRoC ATE of 2.2 cm and tracks all nine TUM-RGBD sequences, achieving 83% lower ATE than DeepFactors and 90% lower ATE than DeepV2D. These results show the accuracy advantage of dense BA with recurrent updates, although the correlation volume and repeated update steps impose substantial memory pressure and make it sensitive to input resolution, sequence length, and hardware constraints. The framework is illustrated in Fig. 12.

Recent solver-level methods further attempt to reduce the computational burden of dense optimization. DPVO [77] replaces dense flow with patch-based correspondence tracking and couples a recurrent update operator with differentiable bundle adjustment. It is evaluated on TartanAir, TUM-RGBD, EuRoC, and ICL-NUIM, where the patch-based design reduces the cost of dense flow while retaining competitive odometry accuracy. DPV-SLAM [78] extends this direction from odometry to SLAM by adding loop-closure mechanisms and global consistency correction. It maintains a small memory overhead of about 5–7 GB,

runs at approximately 1x–4x real-time frame rates on real-world datasets, and achieves accuracy comparable to DROID-SLAM on EuRoC and TartanAir while running about 2.5x faster with lower memory usage. These representative methods indicate different solver-level design choices, ranging from differentiable feature-metric BA to recurrent and patch-based optimization. Their main mechanisms, insertion points, and limitations are summarized in Table 6.

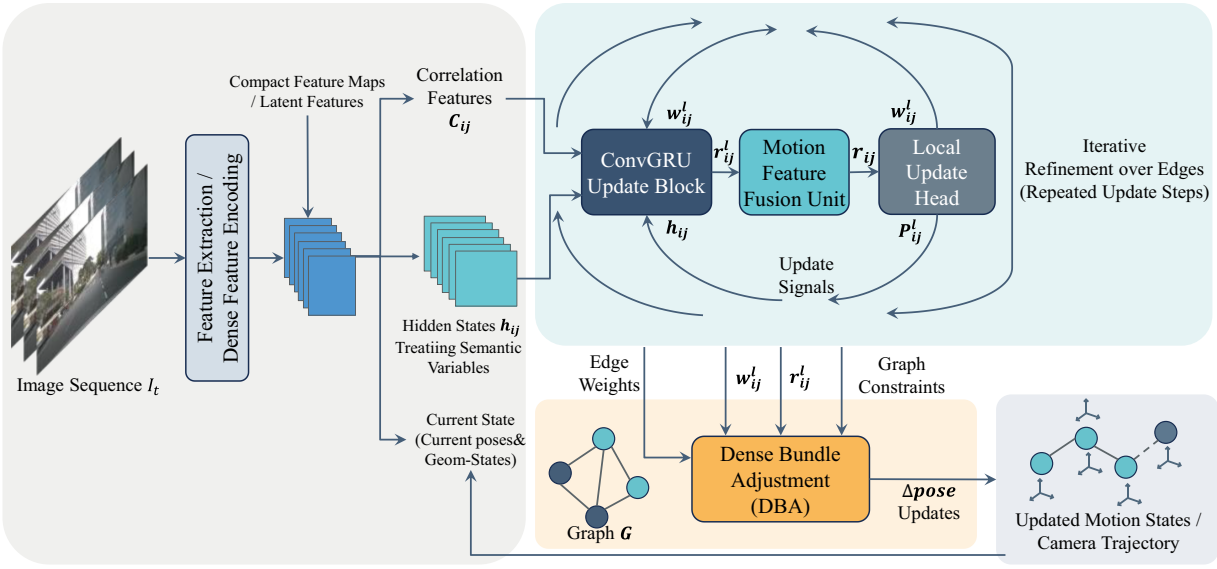

Figure 12: Solver-level interface in DROID-SLAM, where dense correspondence, recurrent updates, edge weighting, and dense bundle adjustment are coupled within a learnable optimization loop.

Table 6: Representative solver-level interfaces in learning-enhanced VSLAM.

Method	Main Solver-Level Mechanism	Geometric Insertion Point	Main Limitation
BA-Net [75]	Differentiable feature-metric BA	Dense SfM and local bundle adjustment	Optimization cost and limited scalability
DeepV2D [76]	Differentiable structure-from-motion	Alternating depth-motion estimation	Iterative inference overhead
DROID-SLAM [22]	Recurrent update with dense BA	Tracking and local bundle adjustment	High memory pressure from correlation volumes
DPVO [77]	Patch-based recurrent update with differentiable BA	Visual odometry	Dependence on patch selection and local tracking stability
DPV-SLAM [78]	Patch-based recurrent update with differentiable BA	Loop-closure-aware monocular SLAM	Dependence on loop-closure reliability and global consistency correction

Overall, solver-level interfaces provide a closer connection between learning objectives and trajectory-level SLAM behavior than observation-level or prior-level interfaces. Their practical risks include numerical instability, difficulty in propagating gradients, high training costs, memory pressure, and reduced

transparency in failure diagnosis. Therefore, deployable solver-level systems usually combine learnable optimization with residual clipping, robust kernels, keyframe-window control, geometric gating, and fallback strategies. These safeguards prevent unstable learned updates from propagating into global trajectory drift.

3.5 Representation-Level Interface

Representation-level interfaces refer to methods in which learning primarily changes the form, update rule, or rendering mechanism of the map. The choice of map representation determines the type of map constructed by the system, the form of residuals adopted in optimization, and the feasibility of long-term maintenance. Conventional VSLAM systems rely primarily on sparse point clouds and keyframe-based maps, which offer strong interpretability and favorable real-time performance, but remain limited in their ability to support high-quality dense reconstruction and realistic rendering. In recent years, innovations at the representation layer have enabled SLAM systems to progressively acquire capabilities for dense geometry modeling, editable semantic representation, and highly realistic visualization. These advances, however, have also led to substantially increased computational and storage costs.

As illustrated in Fig. 13, TSDF and Surfel are representative dense map representations in RGB-D and multi-camera systems, where depth observations are fused into voxels or surface elements to reconstruct scene geometry. Such representations provide explicit geometric meaning and are well suited for collision checking and navigation, while remaining decoupled from pose-graph optimization. Nevertheless, both TSDF and Surfel are sensitive to the quality of depth measurements, exhibit substantial memory growth as scene scale increases, and struggle to provide stable photometric constraints under severe appearance variations [1,3,79].

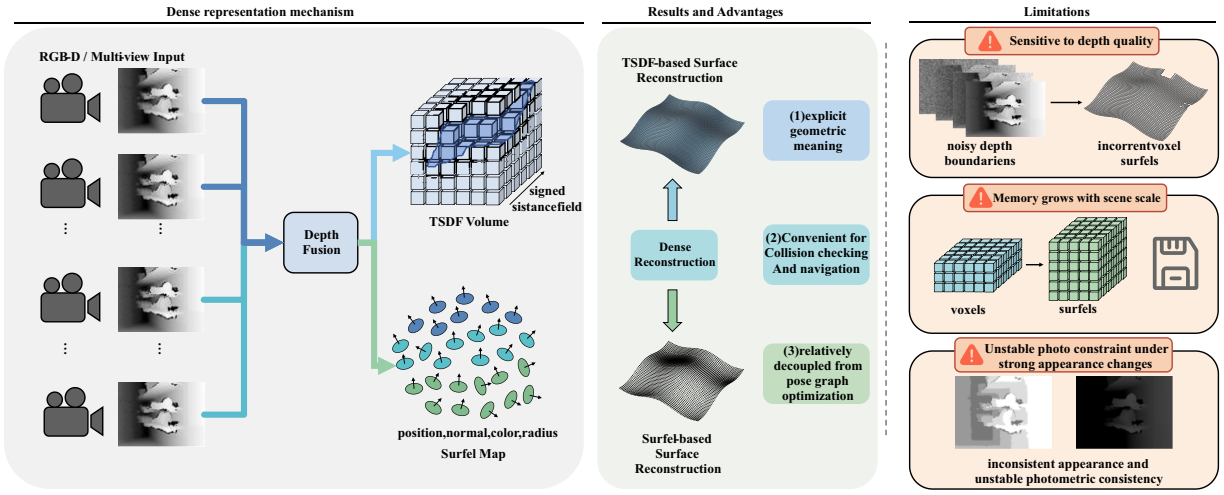

Figure 13: Illustration of TSDF- and Surfel-based dense representations, highlighting their reconstruction characteristics, benefits, and limitations.

NICE-SLAM [26] is a representative neural implicit SLAM system. It introduces scalable implicit scene encoding into dense RGB-D SLAM and represents scene geometry through multi-level neural features. Using depth or rendering consistency for tracking and mapping extends the map from discrete points or voxels to a continuous implicit field. Its Replica evaluation reports an average ATE RMSE of about 1.07 cm across eight scenes, indicating that neural implicit maps can maintain accurate tracking while supporting dense reconstruction. Nevertheless, the implicit representation tends to over-smooth fine details, and its scale bias still requires additional calibration. To meet the demands of real-time operation and large-scale

scenes, Gaussian-SLAM [79] adopted 3D Gaussian Splatting [80] as a dense scene representation in 2023. This framework supports predicting depth or surface normals to strengthen photometric residuals, while introducing uncertainty weighting to enable joint high-fidelity rendering and camera tracking. A notable limitation, however, is its sensitivity to moving objects, which can easily induce drift in environments with heavy occlusion.

Since 2022, neural implicit representations have become a dominant direction in dense SLAM reconstruction. NGEL-SLAM [81] proposed a globally consistent and low-latency SLAM framework based on neural implicit representations, in which geometry and appearance are encoded jointly by continuous functions. By driving coupled tracking and mapping optimization through rendering error, this framework offers clear advantages in dense reconstruction quality and renderability, while naturally accommodating semantics and uncertainty. On TUM RGB-D, it reports ATE-RMSE values of 1.5, 0.5, and 1.0 cm on fr1/desk, fr2/xyz, and fr3/office, compared with 2.7, 1.8, and 3.0 cm for NICE-SLAM. On ScanNet, it achieves an average ATE-RMSE of 7.44 cm, slightly lower than 7.50 cm for ESLAM and 9.63 cm for NICE-SLAM. Its mapping iteration time is 14 ms, compared with 130 ms for NICE-SLAM and 19 ms for ESLAM, supporting its low-latency design. Nevertheless, online training and incremental updates impose substantial demands on computation and GPU memory, and the model size can easily become difficult to control in large-scale environments. To alleviate the burden of submap management, NeRF-SLAM [27] introduced keyframe-window optimization in 2023, restricting implicit mapping to local windows. This design significantly reduced real-time computational cost while preserving high-fidelity reconstruction capability. However, it remains highly sensitive to variations in exposure, moving objects, and reflective surfaces, which can readily cause inconsistencies or drift in appearance.

To achieve a better balance between real-time efficiency and rendering quality, SplaTAM [28] proposed 3D Gaussian Splatting as an efficient alternative in 2024. This method uses Gaussian primitives with color and covariance as the basic representation, enabling fine-grained appearance modeling at higher rendering rates while supporting gradient-based optimization. It uses Gaussian primitives with color and covariance as the basic representation. It achieves up to 2x better performance in camera pose estimation, map construction, and novel-view synthesis compared with existing dense SLAM methods. Similarly, in systems such as GS-SLAM [29], camera poses and Gaussian parameters can be jointly optimized through rendering error, thereby supporting integrated dense mapping and tracking. In outdoor systems such as LiV-GS, LiDAR-based geometric constraints can further strengthen scale observability and structural stability. However, these methods still face notable challenges in online updating and resource management. In particular, Gaussian primitives may grow rapidly as the scene scale increases, and moving objects can introduce Gaussian contamination, necessitating collaborative pruning or hierarchical maintenance with dynamic semantic mechanisms.

In 2025, 2DGS-SLAM [82] and GauS-SLAM [83] further addressed the aforementioned growth and contamination issues by adopting a 2D Gaussian global-consistency framework and Gaussian surfels for dense reconstruction, respectively, thereby improving depth-rendering consistency and loop-closure stability. Gaussian-plus-SDF SLAM [84] combines an SDF geometry backbone with residual Gaussian appearance refinement, achieving 150+ fps reconstruction on real-world scenes and 250+ fps on Replica, with about 50% fewer Gaussians and 75% fewer optimization iterations. WildGS-SLAM [31] introduces uncertainty-aware geometric mapping for monocular dynamic scenes; on the Wild-SLAM MoCap dataset, it reports an average ATE RMSE of 0.46 cm and PSNR/SSIM/LPIPS of 20.59/0.783/0.209, outperforming Splat-SLAM, which reports 8.71 cm and 17.23/0.699/0.346. Collectively, these variants have advanced the practical deployment of 3DGS in SLAM. Even so, an effective system design still generally requires a decoupled threading strategy

in which localization remains the primary task and mapping is executed in the background, together with Gaussian scarification and lifecycle management to control the overhead of online updates.

These representation-level methods improve dense reconstruction and map expressiveness, while their deployment in complete VSLAM systems further requires online scheduling, memory control, dynamic-object handling, and map lifecycle management. These system-level issues are discussed in Section 3.6.

3.6 System-Level Integration Interface

The system-level integration interface refers to complete learning-enhanced VSLAM architectures in which multiple learned modules are coordinated with geometric components across tracking, mapping, relocalization, and loop closure. This interface focuses on the scheduling, coupling, and constraint mechanisms that connect observation-level, constraint/prior/weight-level, solver-level, and representation-level modules within a complete SLAM pipeline.

The first common integration pattern is front-end learning enhancement with a geometric back end. Learned features, matches, semantic masks, or visual place recognition candidates improve the quality of observations, while PnP, BA, PGO, and geometric verification remain responsible for final state estimation. The quantitative results discussed in Section 3.2 show the system motivation of this pattern: SuperGlue improves outdoor pose AUC@5°/10°/20° from 9.80/18.99/30.88 to 34.18/50.32/64.16, LightGlue reaches 150 FPS at 1024 key points and 50 FPS at 4096 key points on an RTX 3080, and Efficient LoFTR provides about 2.5x speedup over LoFTR. These results indicate that front-end learning is effective when geometric constraints still filter its confidence scores and correspondences. Otherwise, overconfident matches, unstable masks, or incorrect retrieval candidates may propagate into wrong constraints or false loop closures.

A second integration pattern is optimization-in-the-loop integration. Differentiable solvers or learnable iterative update modules are embedded into tracking or local mapping, allowing learning objectives to influence pose refinement more directly. The results in Section 3.4 illustrate the trade-off: BA-Net reduces ScanNet rotation and translation errors to 1.018° and 3.39 cm, compared with 3.791° and 10.81 cm for DeMoN, while DROID-SLAM and DPVO further connect recurrent updates with dense or patch-based bundle adjustment. Such systems improve trajectory-level consistency, but their repeated update steps and optimization windows increase memory pressure, training costs, and numerical stability requirements. Deployable systems therefore retain keyframe-window control, robust kernels, residual clipping, geometric gating, and fallback mechanisms.

A third integration pattern is neural-map-centered integration. Tracking, mapping, and rendering are coupled through neural implicit fields, SDFs, or Gaussian primitives. The representation-level results in Section 3.5 show why this pattern is attractive: NICE-SLAM reaches about 1.07 cm ATE RMSE on Replica, NGEL-SLAM reduces mapping iteration time to 14 ms compared with 130 ms for NICE-SLAM, and GS-SLAM reports rendering speed of about 400 FPS. These gains are achieved by coordinating localization, dense reconstruction, and rendering. Still, deployment remains constrained by the cost of online updates, GPU memory consumption, map growth, moving-object contamination, and long-term map maintenance. In practice, localization is usually kept on the real-time path, while mapping, semantic updating, Gaussian pruning, and lifecycle maintenance are executed in the background or on low-frequency threads.

Gaussian-based SLAM systems further illustrate the importance of lifecycle management. As scene scale increases, Gaussian primitives may grow rapidly, and moving objects may contaminate the map if they are fused without semantic or geometric filtering. Practical systems therefore introduce pruning, submap organization, dynamic-static decoupled mapping, and cross-temporal consistency checking. For example, BDGS-SLAM [85] uses Bayesian dynamic/static verification and multi-view probability updating; on the

TUM dataset, it reduces the average ATE from 0.0441 m with DG-SLAM to 0.0247 m, while increasing PSNR from 22.34 to 23.54 dB, increasing SSIM from 0.799 to 0.900, and reducing LPIPS from 0.186 to 0.159. Its tracking and mapping times are 81.4 and 473.4 ms per frame, respectively, giving an average total runtime of 565.3 ms per frame. This example shows that tighter system coupling can improve robustness and rendering quality, but background mapping and semantic inference still dominate runtime.

Recent systems such as BDGS-SLAM [85], HI-SLAM2 [86], SEGS-SLAM [87], GigaSLAM [88], WildGS-SLAM [31], RGD-SLAM [89], and CAD-SLAM [90] introduce mechanisms such as visual-inertial constraints, probabilistic modeling, hierarchical subgraphs, dynamic-static decoupled mapping, low-frequency background updates, and lifecycle management. These mechanisms aim to control map growth, reduce resource contention, and maintain long-term consistency during large-scale or dynamic operations.

From a system perspective, learning-enhanced VSLAM should be evaluated by the stability of the complete pipeline. Important criteria include bounded latency, stable tracking, reliable relocalization, controlled map growth, calibrated uncertainty, and robustness under cross-domain and dynamic conditions. Learned modules provide stronger observations, priors, optimization updates, and map representations. Geometric verification, uncertainty calibration, keyframe management, thread scheduling, and fallback strategies then prevent local learning errors from being amplified into global trajectory drift or map corruption.

4 Frontiers, Open Challenges, and Future Directions

4.1 Foundation Models and Vision-Language-Enhanced VSLAM

Recent foundation models and vision-language models have introduced a new direction for learning-enhanced VSLAM. Unlike conventional CNN- or RNN-based modules that are usually trained for specific tasks such as feature extraction, depth estimation, or semantic segmentation, foundation models are pretrained on large-scale data. They can provide more general visual, geometric, and semantic priors. Typical examples include visual foundation models such as DINOv2, SAM, MobileSAM, and Depth Anything, as well as vision-language models such as CLIP-based open-vocabulary perception models. Their main value for VSLAM lies in open-vocabulary semantic understanding, category-agnostic dynamic-object segmentation, language-conditioned place recognition, semantic loop closure, and quarriable map construction.

At the perception level, foundation models can improve the robustness and generality of learned observations. For example, Depth Anything V2 [91] provides a family of monocular depth foundation models with parameter scales ranging from 25M to 1.3B, enabling a balance between depth quality and deployment cost across different hardware configurations. SAM [92] and MobileSAM [93] can provide category-agnostic segmentation masks, which are useful for suppressing unknown dynamic objects that are difficult to handle with closed-set detectors. In dynamic SLAM and Gaussian-map maintenance, such masks can serve as constraint-, prior-, or weight-level cues to prevent unreliable regions from entering pose estimation or map updates.

At the mapping and semantic understanding level, vision-language models enable open-vocabulary 3D representations. OpenScene [94] maps 3D points into a CLIP-aligned feature space and supports zero-shot 3D semantic segmentation with arbitrary text queries. VLMaps [95] further fuses pretrained visual-language features into spatial maps, allowing natural-language indexing of landmarks and spatial relations. ConceptGraphs [96] and HOV-SG [97] extend this idea to open-vocabulary 3D scene graphs. In particular, HOV-SG reports an 11.69% improvement over ConceptGraphs for object-room-floor queries, a 2.2% advantage for object-room queries, and a 56.1% real-world success rate for long-query object navigation across multiple rooms and floors. These results indicate that vision-language maps can support high-level semantic reasoning beyond conventional geometric maps.

Recent open-vocabulary SLAM systems further attempt to integrate these semantic capabilities into online SLAM pipelines. OVO-SLAM [98] detects and tracks 3D segments from RGB-D frames, describes them using aggregated CLIP features, and integrates the resulting open-vocabulary semantic mapping thread with Gaussian-SLAM. It reports better segmentation metrics and lower computational and memory costs than offline open-vocabulary mapping baselines, while avoiding reliance on ground-truth camera poses or prebuilt scene geometry. Similarly, context-based visual-language place [55] recognition constructs semantic image descriptors via zero-shot, language-driven segmentation, thereby improving robustness to appearance changes without additional task-specific training. These methods suggest that vision-language information can assist loop closure and relocalization by providing semantic context that is more stable than low-level appearance features.

However, foundation models and vision-language models should not be regarded as replacements for geometric SLAM. Their outputs are often semantic or descriptive rather than metric, and they may suffer from hallucination, semantic drift, domain bias, temporal inconsistency, and high computational cost. Open-vocabulary labels or language-conditioned retrieval results do not directly guarantee peripolar consistency, scale consistency, or valid pose-graph constraints. Therefore, a practical integration paradigm is that foundation models propose semantic hypotheses, while geometric SLAM verifies them through metric constraints. For example, VLM-based place candidates should still pass local feature matching and geometric verification before being inserted into PGO, and SAM- or CLIP-based semantic masks should be combined with reprojection residuals, optical flow, or multi-view consistency checks before being used to affect BA or map updates.

From the perspective of the fusion-interface taxonomy, foundation models can appear at multiple interfaces. They can serve as observation-level modules by providing robust features, masks, depth, or semantic descriptors; as constraint/prior/weight-level modules by supplying semantic masks, confidence cues, or object priors; as representation-level modules by enabling open-vocabulary maps and scene graphs; and as system-level modules by supporting language-conditioned relocalization, map query, and task planning. Their future development for VSLAM should focus on geometry-aware foundation models, calibrated semantic confidence, lightweight edge deployment, temporal consistency, and failure-safe integration with classical geometric verification.

4.2 Failure Modes and Reliability of Learned Components

Beyond average accuracy and runtime, learning-enhanced VSLAM should also be evaluated by how learned modules fail under distributional shift, rare motion patterns, dynamic interference, sensor degradation, and long-term environmental changes. Different learned components fail through different mechanisms, but their local errors are often amplified by geometric optimization and system-level feedback. Table 7 summarizes typical failure modes across the five fusion interfaces and corresponding mitigation strategies.

Overall, these failure modes indicate that learned components should be paired with explicit verification rather than trusted as isolated predictors. A failure-aware VSLAM system should record module-level confidence, geometric residuals, rejected hypotheses, fallback triggers, and map-update decisions, enabling detection of distributional shifts before local perception errors accumulate into global trajectory drift or irreversible map corruption.

Table 7: Typical failure modes across the five fusion interfaces.

Fusion Interface	Typical Failure Mode	System-Level Consequences and Mitigation
Observation-level interface	Domain shifts in learned features, matches, depth, or flow.	Mismatches, biased depth, tracking loss; use geometric verification, RANSAC fallback, and multi-view checks.
Constraint/prior/weight-level interface	Miscalibrated confidence, uncertainty, masks, or priors.	Biased residual weighting; use calibration, variance clipping, Mahalanobis checks, and χ^2 gating.
Solver-level interface	Poor conditioning, unstable gradients, or out-of-distribution updates.	Biased convergence or unstable optimization; use damping, bounded updates, and hybrid geometric solvers.
Representation-level interface	Dynamic-object fusion, stale appearance, or uncontrolled map growth.	Map contamination and memory burden; use pruning, submaps, dynamic-static separation, and temporal checks.
System-level integration interface	Error propagation across tracking, mapping, relocalization, and loop closure.	Global drift, false loops, and corrupted maps: use multi-stage gating, fallbacks, logging, and metric verification.

4.3 Practical Challenges and Future Directions for Learning-Enhanced VSLAM

Foundation models and vision-language techniques create new opportunities for learning-enhanced VSLAM, but practical deployment still depends on robustness, uncertainty calibration, real-time efficiency, dynamic map maintenance, and long-term consistency. Research is therefore shifting from proof-of-concept studies toward efficient, stable, and large-scale deployment.

(1) ***Computational resource consumption.*** As VSLAM moves toward engineering-oriented deployment, computational capability and energy efficiency have become hard constraints that directly determine system usability. A practical SLAM system must execute front-end feature extraction and matching, tracking, loop closure search, and back-end optimization in parallel under strict real-time requirements. Once deep neural networks for features, depth, semantics, or novel map representations are introduced, inference latency, GPU memory usage, and power consumption often become critical bottlenecks, especially on embedded platforms such as uncrewed aerial vehicles and mobile robots.

At present, feasible optimization strategies can be considered at three levels: the model level, the system level, and the hardware level. At the model level, network size can be reduced by combining lightweight backbones with pruning, quantization, and distillation. When necessary, confidence output should also be calibrated to avoid weight distortion. At the system level, heavy models are typically placed outside the critical path, for example, in loop closure candidate retrieval or dense mapping, or are activated only on demand. At the same time, redundant computation is reduced through keyframe caching, adaptive resolution, and incremental updates. At the hardware level, GPU/NPU parallelism and operator fusion can significantly reduce edge-side latency, although operator compatibility and cross-platform portability must already be considered during model design. It should also be noted that excessive compression may degrade generalization or distort confidence estimates, whereas cloud-edge collaboration introduces additional risks

to latency and reliability. Therefore, a major future trend lies in the co-design of algorithms and system architectures, with joint optimization targeted toward task-level performance metrics.

In addition, real-time feasibility should be evaluated across the entire SLAM processing workflow, rather than relying solely on a single acceleration component. In particular, tracking latency, mapping or Gaussian-update frequency, rendering speed, memory growth, and the hardware platform should be reported separately. For online robotic deployment, a 30 Hz tracking requirement corresponds to an approximate per-frame tracking budget of 33 ms. A method that renders hundreds of frames per second may still fail to satisfy real-time localization if pose tracking, semantic inference, or map updating exceeds this budget. For example, Gaussian Splatting SLAM [83] reports live operation at about 3 fps, which is far below the typical 30 Hz requirement for high-rate online tracking. GS-SLAM [29] reports rendering at about 400 FPS. Still, this value mainly reflects the efficiency of the differentiable splatting renderer and does not directly indicate the latency of the full tracking–mapping pipeline. Similarly, Gaussian-plus-SDF SLAM [87] reports 150+ fps reconstruction on real-world scenes and 250+ fps on Replica, suggesting strong reconstruction efficiency, but localization latency and update scheduling still need to be reported separately. BDGS-SLAM [88] further illustrates this gap: its tracking and mapping times are 81.4 and 473.4 ms per frame, respectively, corresponding to about 12.3 Hz for tracking and below 2 Hz for the total runtime.

This distinction is also important for interpreting NeRF- and 3DGS-based SLAM results. Rendering FPS, reconstruction FPS, and tracking FPS describe different parts of the pipeline and should not be used interchangeably. A practical neural-map VSLAM system should keep localization on the critical real-time path, execute dense mapping and semantic updates in background or low-frequency threads, and report peak GPU memory usage, map growth rate, and power consumption, together with ATE/RPE. Only under such reporting can the real-time deployability of neural implicit and Gaussian-based SLAM systems be judged fairly.

(2) ***Fusion paradigms.*** The difficulty of integrating deep learning into geometric VSLAM arises primarily from the mismatch between the underlying modeling paradigms of the two. Geometric methods rely on explicit observation models and well-defined optimization objectives, emphasizing interpretable residuals and controllable convergence behavior. Deep models, in contrast, are predominantly data-driven and often lack explicit physical constraints and principled uncertainty modeling. As a consequence, naively injecting network outputs into a geometric pipeline can easily introduce cross-domain bias, which may then be amplified along the optimization chain.

From a mechanistic perspective, effective fusion must address at least three fundamental questions. First, what is the geometric meaning of the learned output? Second, how do its associated errors and uncertainties propagate to pose estimation and map construction? Third, are the training objectives consistent with the closed-loop objectives of SLAM? Without such alignment, individual modules often appear effective in isolation yet provide little or no benefit at the system level. At present, viable fusion paradigms can be broadly classified into four categories. The first treats learned features, depth, or semantic outputs as observations, while retaining geometric verification mechanisms such as RANSAC, epipolar constraints, and bundle adjustment (BA) to suppress outliers and drift. The second uses learned confidence to reweight residuals or further learns key parameters of robust kernels, thereby enlarging the optimization convergence basin under noisy and degraded conditions. The third makes part of the optimization process differentiable and incorporates it into training, so that feature representation and solving procedures can jointly adapt to trajectory-level objectives. The fourth introduces emerging map representations, such as neural implicit fields or 3D Gaussian Splatting (3DGS), to enhance representational capacity. However, such approaches usually require decoupling the tracking and mapping threads at the system level to prevent dense mapping computations from occupying the real-time path and causing latency jitter.

Future research should shift from simple module stacking toward unified objectives and uncertainty-aware modeling. On the one hand, this calls for the development of self-supervised, weakly supervised, and online adaptation strategies to improve cross-domain generalization. On the other hand, it requires making uncertainty explicit, so that learning modules and geometric optimization can be statistically aligned, while introducing interpretable diagnostic tools, such as residual decomposition and failure detection, to satisfy the verifiability requirements of safety-critical applications.

Another particularly important future direction is geometry-aware uncertainty calibration. Instead of directly trusting a predicted confidence map, future systems should align uncertainty with reprojection error, multi-view consistency, optical-flow consistency, and pose-graph residual statistics. During online deployment, calibration should also be monitored under domain shift, because overconfident learned modules may silently convert local perception errors into global trajectory drift. Combining calibrated uncertainty with OOD detection, residual diagnostics, and conservative fallback strategies will be essential for safety-critical and long-term VSLAM applications.

(3) ***Long-term and large-scale operation*****.** In application scenarios such as autonomous driving, inspection robotics, and augmented reality, VSLAM must address the dual challenges of long-term operation and large-scale environmental mapping. Among these, cumulative error remains the most prominent difficulty: local estimation errors continuously accumulate over time, eventually leading to trajectory drift and map distortion. Although loop closure detection and global optimization can partially alleviate these issues, large-scale environments impose substantial computational burdens on place-search operations. At the same time, appearance variation can significantly reduce the reliability of loop closure. Moreover, environmental dynamics further aggravate these challenges. Day-night transitions, weather and seasonal changes, and variations in indoor furniture layout or lighting conditions may all undermine map consistency. Most existing SLAM methods are still built on a static-world assumption and lack effective adaptive updating mechanisms, resulting in degraded long-term accuracy. At the same time, map size increases rapidly with exploration range; city-scale maps may reach tens of gigabytes, making it difficult for conventional optimization methods to converge within a reasonable time budget. Although strategies such as hierarchical management, regional optimization, and distributed mapping have been proposed, maintaining global consistency while avoiding information redundancy remains an open challenge.

To address these issues, current research is exploring hierarchical map structures, sparse optimization, and learning-driven appearance modeling to improve long-term consistency and cross-temporal robustness. Among these directions, cloud-edge collaboration offers a promising framework for map management: edge devices handle local real-time localization, whereas remote servers perform global optimization, thereby enabling an efficient division of labor between local and cloud resources. This paradigm further supports map sharing and collaborative optimization, which is particularly important for multi-robot systems. Looking ahead, future developments are likely to shift toward dynamic maps with self-updating capabilities, combining deep-learning-based appearance modeling with geometric constraints to achieve higher accuracy and reliability in long-term, large-scale operation.

5 Conclusion

This review has examined deep learning-enhanced VSLAM from the perspective of fusion interfaces between learning modules and geometric SLAM pipelines. We organized representative studies by the point at which their outputs enter the SLAM process, including observation-level interfaces, constraint/prior/weight-level interfaces, solver-level interfaces, representation-level interfaces, and system-level integration interfaces. This interface-oriented view clarifies the functional roles, boundary rules,

performance trade-offs, and failure propagation mechanisms of different learning-enhanced VSLAM methods.

Overall, deep learning has improved VSLAM by providing more robust observations, adaptive priors and weights, differentiable optimization mechanisms, expressive neural or Gaussian map representations, and more integrated system-level pipelines. Nevertheless, practical deployment of learning-enhanced VSLAM remains constrained by cross-domain generalization, uncertainty miscalibration, real-time computational burden, interference from dynamic scenes, map contamination, and the lack of standardized evaluation protocols. Recent foundation models and vision-language techniques further expand the semantic and open-vocabulary capabilities of VSLAM. Still, they should be integrated as geometry-verified semantic hypotheses rather than as replacements for metric SLAM constraints.

Future research should therefore focus on reliable, efficient, and interpretable learning-enhanced VSLAM systems. Promising directions include geometry-aware foundation models, calibrated uncertainty estimation, lightweight neural solvers, real-time neural map updating, dynamic and long-term map maintenance, failure-aware fallback mechanisms, and standardized benchmarks that jointly report accuracy, robustness, runtime, memory consumption, and system-level reliability. In this direction, deep learning and geometric SLAM are expected to evolve not as competing paradigms but as complementary components of robust autonomous perception systems.

Acknowledgement: Not applicable.

Funding Statement: This research was funded by the Central Fund for Basic Scientific Research at Central Universities Defense Project, Grant No. 2024CDJGF-053.

Author Contributions: The authors confirm contribution to the paper as follows: Conceptualization, Xiruo Chen; methodology, Xiruo Chen; software, Xiruo Chen; validation, Xiruo Chen; formal analysis, Xiruo Chen and Qi Ouyang; investigation, Xiruo Chen and Yuke Meng; resources, Qi Ouyang; data curation, Xiruo Chen and Sihong Meng; writing—original draft preparation, Xiruo Chen and Sihong Meng; writing—review and editing, Xiruo Chen and Qi Ouyang; visualization, Xiruo Chen and Sihong Meng; supervision, Xiruo Chen; project administration, Xiruo Chen and Sihong Meng; funding acquisition, Qi Ouyang. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: Not applicable.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Mokssit S, Licea DB, Guermah B, Ghogho M. Deep learning techniques for visual SLAM: A survey. IEEE Access. 2023;11(3):20026–50. doi:10.1109/access.2023.3249661.
2. Tourani A, Bavle H, Sanchez-Lopez JL, Voos H. Visual SLAM: What are the current trends and what to expect? Sensors. 2022;22(23):9297. doi:10.3390/s22239297.
3. Favorskaya MN. Deep learning for visual SLAM: The state-of-the-art and future trends. Electronics. 2023;12(9):2006. doi:10.3390/electronics12092006.
4. Zhao L, Chen T, Yuan P, Li X, Chen B. Review of deep learning-based visual SLAM: Types, approaches, and future work. Ind Robot Int J Robot Res Appl. 2026;53(2):348–63. doi:10.1108/ir-04-2025-0137.
5. Li S, Zhang D, Xian Y, Li B, Zhang T, Zhong C. Overview of deep learning application on visual SLAM. Displays. 2022;74(6):102298. doi:10.1016/j.displa.2022.102298.
6. Abaspur Kazerouni I, Fitzgerald L, Dooly G, Toal D. A survey of state-of-the-art on visual SLAM. Expert Syst Appl. 2022;205(6):117734. doi:10.1016/j.eswa.2022.117734.

7. Huang K, Zhang S, Zhang J, Tao D. Event-based simultaneous localization and mapping: A comprehensive survey. arXiv:2304.09793. 2023.
8. Campos C, Elvira R, Rodriguez JJG, M Montiel JM, D Tardos J. ORB-SLAM3: an accurate open-source library for visual, visual-inertial, and multimap SLAM. IEEE Trans Robot. 2021;37(6):1874–90. doi:10.1109/TRO.2021.3075644.
9. TUM. RGB-D SLAM dataset and benchmark [Internet]. [cited 2026 Mar 16]. Available from: https://cvg.cit.tum.de/data/datasets/rgbd-dataset.
10. ICL. ICL-NUIM dataset [Internet]. [cited 2026 Mar 16]. Available from: https://www.doc.ic.ac.uk/~ahanda/VaFRIC/iclnuim.html.
11. ETHZ ASL. EuRoC MAV dataset [Internet]. 2016 [cited 2026 Mar 16]. Available from: https://projects.asl.ethz.ch/datasets/doku.php?id=kmavvisualinertialdatasets.
12. KITTI. The KITTI vision benchmark suite [Internet]. [cited 2026 Mar 16]. Available from: https://www.cvlibs.net/datasets/kitti/.
13. Oxford RobotCar dataset [Internet]. [cited 2026 Mar 16]. Available from: https://robotcar-dataset.robots.ox.ac.uk/.
14. Newer college dataset [Internet]. [cited 2026 Mar 16]. Available from: https://ori-drs.github.io/newer-college-dataset/download/.
15. TartanAir: A dataset to push the limits of visual SLAM [Internet]. [cited 2026 Mar 16]. Available from: https://theairlab.org/tartanair-dataset/.
16. RGB-D Dataset 7-Scenes [Internet]. 2013 [cited 2026 Mar 16]. Available from: https://www.microsoft.com/en-us/research/project/rgb-d-dataset-7-scenes/.
17. ScanNet [Internet]. [cited 2026 Mar 16]. Available from: https://github.com/ScanNet/ScanNet.
18. DeTone D, Malisiewicz T, Rabinovich A. SuperPoint: Self-supervised interest point detection and description. arXiv:1712.07629. 2017.
19. Sarlin PE, DeTone D, Malisiewicz T, Rabinovich A. SuperGlue: Learning feature matching with graph neural networks. In: Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2020 Jun 13–19; Seattle, WA, USA. New York, NY, USA: IEEE; 2020. p. 4937–46. doi:10.1109/cvpr42600.2020.00499.
20. Sun J, Shen Z, Wang Y, Bao H, Zhou X. LoFTR: detector-free local feature matching with transformers. In: Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2021 Jun 20–25; Nashville, TN, USA. New York, NY, USA: IEEE; 2021. p. 8918–27. doi:10.1109/CVPR46437.2021.00881.
21. Yang N, Wang R, Stückler J, Cremers D. Deep virtual stereo odometry: leveraging deep depth prediction for monocular direct sparse odometry. In: Proceedings of the European Conference on Computer Vision (ECCV); 2018 Sep 8–14; Munich, Germany. p. 817–33.
22. Teed Z, Deng J. DROID-SLAM: deep visual SLAM for monocular, stereo, and RGB-D cameras. Adv Neural Inf Process Syst. 2021;34:16558–69.
23. Mildenhall B, Srinivasan PP, Tancik M, Barron JT, Ramamoorthi R, Ng R. NeRF: Representing scenes as neural radiance fields for view synthesis. In: Computer Vision—ECCV 2020. Cham, Switzerland: Springer International Publishing; 2020. p. 405–21. doi:10.1007/978-3-030-58452-8_24.
24. Park JJ, Florence P, Straub J, Newcombe R, Lovegrove S. DeepSDF: Learning continuous signed distance functions for shape representation. In: Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2019 Jun 15–20; Long Beach, CA, USA. New York, NY, USA: IEEE; 2020. p. 165–74. doi:10.1109/CVPR.2019.00025.
25. Kerbl B, Kopanas G, Leimkuehler T, Drettakis G. 3D Gaussian splatting for real-time radiance field rendering. ACM Trans Graph. 2023;42(4):1–14. doi:10.1145/3592433.
26. Zhu Z, Peng S, Larsson V, Xu W, Bao H, Cui Z, et al. NICE-SLAM: Neural implicit scalable encoding for SLAM. In: Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2022 Jun 18–24; New Orleans, LA, USA. New York, NY, USA: IEEE; 2022. p. 12776–86. doi:10.1109/CVPR52688.2022.01245.

27. Rosinol A, Leonard JJ, Carlone L. NeRF-SLAM: Real-time dense monocular SLAM with neural radiance fields. In: Proceedings of the 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS); 2023 Oct 1–5; Detroit, MI, USA. New York, NY, USA: IEEE; 2023. p. 3437–44. doi:10.1109/IROS55552.2023.10341922.
28. Keetha N, Karhade J, Jatavallabhula KM, Yang G, Scherer S, Ramanan D, et al. SplaTAM: Splat, track and map 3D Gaussians for dense RGB-D SLAM. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2024 Jun 17–21; Seattle, WA, USA. p. 21357–66.
29. Yan C, Qu D, Xu D, Zhao B, Wang Z, Wang D, et al. GS-SLAM: dense visual SLAM with 3D Gaussian splatting. In: Proceedings of the 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2024 Jun 16–22; Seattle, WA, USA. New York, NY, USA: IEEE; 2024. p. 19595–604. doi:10.1109/CVPR52733.2024.01853.
30. Zhu Z, Zhang W, Li M, Haala N, Pollefeys M, Barath D. VIGS-SLAM: Visual inertial Gaussian splatting SLAM. arXiv:2512.02293. 2025.
31. Zheng J, Zhu Z, Bieri V, Pollefeys M, Peng S, Armeni I. WildGS-SLAM: monocular Gaussian splatting SLAM in dynamic environments. In: Proceedings of the 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2025 Jun 10–17; Nashville, TN, USA. New York, NY, USA: IEEE; 2025. p. 11461–71. doi:10.1109/CVPR52734.2025.01070.
32. Peng Q, Xiang Z, Fan Y, Zhao T, Zhao X. RWT-SLAM: Robust visual SLAM for highly weak-textured environments. arXiv:2207.03539. 2022.
33. Wang Q, Zhang J, Yang K, Peng K, Stiefelhagen R. MatchFormer: Interleaving attention in Transformers for Feature matching. In: Computer Vision—ACCV 2022. Cham, Switzerland: Springer; 2023. p. 256–73. doi:10.1007/978-3-031-26313-2_16.
34. Chen H, Luo Z, Zhou L, Tian Y, Zhen M, Fang T, et al. ASpanFormer: Detector-free image matching with adaptive span transformer. In: Computer Vision—ECCV 2022. Cham, Switzerland: Springer Nature; 2022. p. 20–36. doi:10.1007/978-3-031-19824-3_2.
35. Zhao X, Wu X, Chen W, Chen PCY, Xu Q, Li Z. ALIKED: A lighter keypoint and descriptor extraction network via deformable transformation. IEEE Trans Instrum Meas. 2023;72:5014016. doi:10.1109/TIM.2023.3271000.
36. Lindenberger P, Sarlin PE, Pollefeys M. LightGlue: local feature matching at light speed. In: Proceedings of the 2023 IEEE/CVF International Conference on Computer Vision (ICCV); 2023 Oct 1–6; Paris, France. New York, NY, USA: IEEE; 2024. p. 17581–92. doi:10.1109/ICCV51070.2023.01616.
37. Berton G, Trivigno G, Caputo B, Masone C. EigenPlaces: training viewpoint robust models for visual place recognition. In: Proceedings of the 2023 IEEE/CVF International Conference on Computer Vision (ICCV); 2023 Oct 1–6; Paris, France. New York, NY, USA: IEEE; 2024. p. 11046–56. doi:10.1109/ICCV51070.2023.01017.
38. Edstedt J, Sun Q, Bökman G, Wadenbäck M, Felsberg M. RoMa: robust dense feature matching. In: Proceedings of the 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2024 Jun 16–22; Seattle, WA, USA. New York, NY, USA: IEEE; 2024. p. 19790–800. doi:10.1109/CVPR52733.2024.01871.
39. Wang Y, He X, Peng S, Tan D, Zhou X. Efficient LoFTR: semi-dense local feature matching with sparse-like speed. In: Proceedings of the 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2024 Jun 16–22; Seattle, WA, USA. New York, NY, USA: IEEE; 2024. p. 21666–75. doi:10.1109/CVPR52733.2024.02047.
40. Hausler S, Garg S, Xu M, Milford M, Fischer T. Patch-NetVLAD: multi-scale fusion of locally-global descriptors for place recognition. In: Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2021 Jun 20–25; Nashville, TN, USA. New York, NY, USA: IEEE; 2021. p. 14136–47. doi:10.1109/cvpr46437.2021.01392.
41. Wang R, Shen Y, Zuo W, Zhou S, Zheng N. TransVPR: transformer-based place recognition with multi-level attention aggregation. In: Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2022 Jun 18–24; New Orleans, LA, USA. New York, NY, USA: IEEE; 2022. p. 13638–47. doi:10.1109/CVPR52688.2022.01328.
42. Izquierdo S, Civera J. Optimal transport aggregation for visual place recognition. In: Proceedings of the 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2024 Jun 16–22; Seattle, WA, USA. New York, NY, USA: IEEE; 2024. p. 17658–68. doi:10.1109/CVPR52733.2024.01672.

43. Berton G, Masone C, Caputo B. Rethinking visual geo-localization for large-scale applications. In: Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2022 Jun 18–24; New Orleans, LA, USA. New York, NY, USA: IEEE; 2022. p. 4868–78. doi:10.1109/CVPR52688.2022.00483.
44. Khaliq A, Xu M, Hausler S, Milford M, Garg S. VLAD-BuFF: burst-aware fast feature aggregation for Visual place recognition. In: Computer Vision—ECCV 2024. Cham, Switzerland: Springer; 2025. p. 447–66. doi:10.1007/978-3-031-72784-9_25.
45. Wang C, Min S. Spatial-frequency: dual-domain aggregation for visual place recognition. In: Proceedings of the 36th British Machine Vision Conference 2025; 2025 Nov 24–27; Sheffield, UK. p. 4416–25.
46. Hausler S, Moghadam P. Pair-VPR: place-aware pre-training and contrastive pair classification for visual place recognition with vision transformers. IEEE Robot Autom Lett. 2025;10(4):4013–20. doi:10.1109/LRA.2025.3546512.
47. Arshad S, Park TH. SVS-VPR: a semantic visual and spatial information-based hierarchical visual place recognition for autonomous navigation in challenging environmental conditions. Sensors. 2024;24(3):906. doi:10.3390/s24030906.
48. Zhao X, Zhang Y, Ning J, Wang Z, Li K, Zou D. Semantic boundary constrained network for visual place recognition under adverse conditions. IEEE Trans Intell Transp Syst. 2025;26(8):11823–34. doi:10.1109/TITS.2025.3576106.
49. Hu Z, Liao L, Lin W. SRD2-VPR: semantics-enforced feature aggregation with query rejection for visual place recognition. IEEE Trans Circuits Syst Video Technol. 2026;36(5):6934–49. doi:10.1109/TCSVT.2026.3651681.
50. Wang C, Li Y, Min S, Chen X. Robust visual place recognition with adaptive deformable token aggregation. Comput Graph. 2025;131(3):104342. doi:10.1016/j.cag.2025.104342.
51. Lin Y, Evans N. Semantic-enhanced cross-modal place recognition for robust robot localization. arXiv:2509.13474. 2025.
52. Wang L, Lan C, Wu B, Yao F, Wei Z, Gao T, et al. High-level adaptive feature enhancement and attention mask-guided aggregation for visual place recognition. Knowl Based Syst. 2026;336(8):115285. doi:10.1016/j.knosys.2026.115285.
53. Joseph T, Fischer T, Milford M. Ensemble-based event camera place recognition under varying illumination. arXiv:2509.01968. 2025.
54. Woo S, Kim SW. Context-based visual-language place recognition. arXiv:2410.19341. 2024.
55. Wang W, Hu Y, Scherer SA. TartanVO: a generalizable learning-based VO. In: Proceedings of the 2020 Conference on Robot Learning; 202 Nov 16–18; Online. p. 1761–72.
56. Zheng W, Ou L, He J, Zhou L, Yu X, Wei Y. UP-SLAM: adaptively structured Gaussian SLAM with uncertainty prediction in dynamic environments. arXiv:2505.22335. 2025.
57. Gao S, Zhang M, Gao X, Zhang D. DMS-SLAM: semantic visual SLAM based on deep mask segmentation in dynamic environments. Meas Sci Technol. 2025;36(4):046311. doi:10.1088/1361-6501/adc1f1.
58. Zhou K, Yu Z, Zhou X, Tan P, Yin Y, Luo H. ADEmono-SLAM: absolute depth estimation for monocular visual simultaneous localization and mapping in complex environments. Electronics. 2025;14(20):4126. doi:10.3390/electronics14204126.
59. Kazi K, Kalhoro AN, Memon F, Memon AR, Iqbal M. Beyond handcrafted features: a deep learning framework for optical flow and SLAM. J Imaging. 2025;11(5):155. doi:10.3390/jimaging11050155.
60. Wei W, Xia C, Han J. DI-SLAM: a Real-Time enhanced RGB-D SLAM for dynamic indoor environments. Appl Sci. 2025;15(8):4446. doi:10.3390/app15084446.
61. Bescos B, Fácil JM, Civera J, Neira J. DynaSLAM: tracking, mapping, and inpainting in dynamic scenes. IEEE Robot Autom Lett. 2018;3(4):4076–83. doi:10.1109/LRA.2018.2860039.
62. Bescos B, Cadena C, Neira J. Empty cities: a dynamic-object-invariant space for visual SLAM. IEEE Trans Robot. 2021;37(2):433–51. doi:10.1109/TRO.2020.3031267.
63. Hu X, Lang J. DOE-SLAM: dynamic object enhanced visual SLAM. Sensors. 2021;21(9):3091. doi:10.3390/s21093091.
64. Fan Y, Zhang Q, Tang Y, Liu S, Han H. Blitz-SLAM: a semantic SLAM in dynamic environments. Pattern Recognit. 2022;121(2):108225. doi:10.1016/j.patcog.2021.108225.

65. Qin Y, Mei T, Gao Z, Lin Z, Song W, Zhao X. RGB-D SLAM in dynamic environments with multilevel semantic mapping. J Intell Robot Syst. 2022;105(4):90. doi:10.1007/s10846-022-01697-y.
66. Zhang XY, Abd Rahman AH, Qamar F. Semantic visual simultaneous localization and mapping (SLAM) using deep learning for dynamic scenes. PeerJ Comput Sci. 2023;9(12):e1628. doi:10.7717/peerj-cs.1628.
67. Zhu S, Wang G, Blum H, Liu J, Song L, Pollefeys M, et al. SNI-SLAM: semantic neural implicit SLAM. In: Proceedings of the 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2024 Jun 16–22; Seattle, WA, USA. New York, NY, USA: IEEE; 2024. p. 21167–77. doi:10.1109/CVPR52733.2024.02000.
68. Jiang M, Kim C, Chen Z, Li F. GS4: generalizable sparse splatting semantic SLAM. arXiv:2506.06517. 2025.
69. Muslim MT, Selamat H, Aburaya A. YOSO-SLAM: A real-time object visual SLAM for dynamic scenes with semantic three-dimensional mapping. Arab J Sci Eng. 2026;51(6):8839–58. doi:10.1007/s13369-025-10840-4.
70. Galindo C, Saffiotti A, Coradeschi S, Buschka P, Fernandez-Madrigal JA, Gonzalez J. Multi-hierarchical semantic maps for mobile robotics. In: Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems; 2005 Aug 2–6; Edmonton, AB, Canada. New York, NY, USA: IEEE; 2005. p. 2278–83. doi:10.1109/IROS.2005.1545511.
71. Liu H, Xu G, Liu B, Li Y, Yang S, Tang J, et al. A real time LiDAR-Visual-Inertial object level semantic SLAM for forest environments. ISPRS J Photogramm Remote Sens. 2025;219:71–90. doi:10.1016/j.isprsjprs.2024.11.013.
72. Wang N, Lu H, Zheng Z, Liu YH, Chen X. Leveraging semantic graphs for efficient and robust LiDAR SLAM. In: Proceedings of the 2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS); 2025 Oct 19–25; Hangzhou, China. New York, NY, USA: IEEE; 2025. p. 1614–21. doi:10.1109/IROS60139.2025.11246943.
73. Liu Y, Wang Y, Zhang H, Li Q. Geometric constraints and semantic optimization SLAM algorithm for dynamic scenarios. Sci Rep. 2025;15(1):31864. doi:10.1038/s41598-025-16714-x.
74. Gao R, Qi Y. Monocular object-level SLAM enhanced by joint semantic segmentation and depth estimation. Sensors. 2025;25(7):2110. doi:10.3390/s25072110.
75. Tang C, Tan P. BA-Net: dense bundle adjustment networks. In: Proceedings of the International Conference on Learning Representations (ICLR); 2019 May 6–9; New Orleans, LA, USA; 2019. p. 1–17.
76. Teed Z, Deng J. DeepV2D: video to depth with differentiable structure from motion. In: Proceedings of the International Conference on Learning Representations (ICLR 2020); 2020 Apr 30; Addis Ababa, Ethiopia.
77. Teed Z, Lipson L, Deng J. Deep patch visual odometry. Adv Neural Inf Process Syst. 2023;36:39033–51. doi:10.52202/075280-1696.
78. Lipson L, Teed Z, Deng J. Deep patch visual SLAM. In: European Conference on Computer Vision (ECCV). Cham, Switzerland: Springer Nature; 2024. p. 424–40. doi:10.1007/978-3-031-72627-9_24.
79. Yugay V, Li Y, Gevers T, Oswald MR. Gaussian-SLAM: photo-realistic dense SLAM with Gaussian splatting. arXiv:2312.10070. 2023.
80. Matsuki H, Murai R, Kelly PHJ, Davison AJ. Gaussian splatting SLAM. In: Proceedings of the 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2024 Jun 16–22; Seattle, WA, USA. New York, NY, USA: IEEE; 2024. p. 18039–48. doi:10.1109/CVPR52733.2024.01708.
81. Mao Y, Yu X, Wang K, Wang Y, Xiong R, Liao Y. NGEL-SLAM: neural implicit representation-based global consistent low-latency SLAM system. arXiv:2311.09525. 2023.
82. Zhong X, Pan Y, Jin L, Popović M, Behley J, Stachniss C. Globally consistent RGB-D SLAM with 2D Gaussian splatting. arXiv:2506.00970. 2025.
83. Su Y, Chen L, Zhang K, Zhao Z, Hou C, Yu Z. GauS-SLAM: Dense RGB-D SLAM with Gaussian surfels. arXiv:2505.01934. 2025.
84. Peng Z, Zhou K, Shao T. Gaussian-plus-SDF SLAM: High-fidelity 3D reconstruction at 150+ fps. Comput Vis Medium. 2025;11(6):1195–208. doi:10.26599/CVM.2025.9450513.
85. Yang T, Wei S, Nan J, Li M, Li M. BDGS-SLAM: A probabilistic 3D Gaussian splatting framework for robust SLAM in dynamic environments. Sensors. 2025;25(21):6641. doi:10.3390/s25216641.
86. Zhang W, Cheng Q, Skuddis D, Zeller N, Cremers D, Haala N. HI-SLAM2: geometry-aware Gaussian SLAM for fast monocular scene reconstruction. IEEE Trans Robot. 2025;41:6478–93. doi:10.1109/TRO.2025.3626627.

87. Wen T, Liu Z, Fang Y. Segs-slam: structure-enhanced 3D Gaussian splatting slam with appearance embedding. In: Proceedings of the 2025 IEEE/CVF International Conference on Computer Vision (ICCV); 2025 Oct 19–25; Honolulu, HI, USA. New York, NY, USA: IEEE; 2026. p. 28103–13. doi:10.1109/ICCV51701.2025.02609.
88. Deng K, Zhang Y, Yang J, Xie J. GigaSLAM: Large-scale monocular SLAM with hierarchical Gaussian splats. arXiv:2503.08071. 2025.
89. Wang H, Shou Y, Shen L, Li S, Cao Y. RGD-SLAM: robust Gaussian splatting SLAM for dynamic environments. Pattern Recognit. 2026;175(9):113071. doi:10.1016/j.patcog.2026.113071.
90. Wu W, Su C, Zhu S, Deng T, Jiao J, Wang G, et al. CAD-SLAM: Consistency-aware dynamic SLAM with dynamic-static decoupled mapping. arXiv:2505.19420. 2025.
91. Yang L, Kang B, Huang Z, Zhao Z, Xu X, Feng J, et al. Depth anything V2. In: Proceedings of the Advances in Neural Information Processing Systems 37; 2024 Dec 10–15; Vancouver, BC, Canada. San Diego, CA, USA: Neural Information Processing Systems Foundation, Inc.; 2024. p. 21875–911. doi:10.52202/079017-0688.
92. Kirillov A, Mintun E, Ravi N, Mao H, Rolland C, Gustafson L, et al. Segment anything. In: Proceedings of the 2023 IEEE/CVF International Conference on Computer Vision (ICCV); 2023 Oct 1–6; Paris, France. New York, NY, USA: IEEE; 2024. p. 3992–4003. doi:10.1109/ICCV51070.2023.00371.
93. Zhang C, Han D, Qiao Y, Kim JU, Bae SH, Lee S, et al. Faster segment anything: Towards lightweight SAM for mobile applications. arXiv:2306.14289. 2023.
94. Peng S, Genova K, Jiang C, Tagliasacchi A, Pollefeys M, Funkhouser T. OpenScene: 3D scene understanding with open vocabularies. In: Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2023 Jun 17–24; Vancouver, BC, Canada. New York, NY, USA: IEEE; 2023. p. 815–24. doi:10.1109/cvpr52729.2023.00085.
95. Huang C, Mees O, Zeng A, Burgard W. Visual language maps for robot navigation. In: Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA); 2023 May 29–Jun 2; London, UK. New York, NY, USA: IEEE; 2023. p. 10608–15. doi:10.1109/ICRA48891.2023.10160969.
96. Gu Q, Kuwajerwala A, Morin S, Jatavallabhula KM, Sen B, Agarwal A, et al. ConceptGraphs: open-vocabulary 3D scene graphs for perception and planning. In: Proceedings of the 2024 IEEE International Conference on Robotics and Automation (ICRA); 2024 May 13–17; Yokohama, Japan. New York, NY, USA: IEEE; 2024. p. 5021–8. doi:10.1109/ICRA57147.2024.10610243.
97. Werby A, Huang C, Büchner M, Valada A, Burgard W. Hierarchical open-vocabulary 3D scene graphs for language-grounded robot navigation. In: Proceedings of the First Workshop on Vision-Language Models for Navigation and Manipulation at ICRA 2024; 2024 May 13; Yokohama, Japan.
98. Berriel-Martins T, Oswald MR, Civera J. OVO-SLAM: Open-vocabulary online simultaneous localization and mapping. arXiv:2411.15043. 2024.