SSRG International Journal of Mechanical Engineering
ISSN: 2348-8360/ https://doi.org/10.14445/23488360/IJME-V13I7P118

Volume 13 Issue 7, 210-236, July 2026

Review Article

Review of Visual SLAM: From Geometry to Learning with Challenges and Future Directions

Mahesh Kamble[1*], Deepak Watvisave[2], Omkar Watvisave[3], Dattaray Kamble[4]

[1]*Department of Mechanical Engineering, PVG´s College of Engineering, Technology and Management, Pune, Maharashtra, India.*
[2]*Department Mechanical Engineering, Cummins College of Engineering for Women, Pune, Maharashtra, India.*
[3]*Department of Information Technology, Arizona State University, Tempe, AZ, United States.*
[4]*Department Mechanical Engineering, ABMSP´s, Anantrao Pawar College of Engineering & Research, Pune, Maharashtra, India.*

**Corresponding Author : mmk_mech@pvgcoet.ac.in*

Received: 09 January 2026 Revised: 25 March 2026 Accepted: 13 July 2026 Published: 30 July 2026

Abstract *- In recent decades, visual SLAM has evolved enormously from geometric approaches to learning-based approaches. There are various algorithms that are used for a variety of scenarios with different complexity levels. Each of these algorithms has its own strengths and limitations. It becomes difficult to select a visual SLAM algorithm for a particular scenario. This article provides the evolution of front-end architecture of visual SLAM from pure geometric methods to learned features, deep optical flow, semantic segmentation, object-level SLAM and differentiable rendering. It then provides the advancement in backend from filtering techniques to optimization methods, enhancing loop closure, outlier methods, fusion of sensors and learning enhanced backends. This article also showcases the strengths, weaknesses, and use cases of algorithms and their comparison across varied environmental conditions and computational requirements. The paper provides information about the trade-off between some performance parameters along with a few selection guidelines. Towards the end, this article provides major challenges in the implementation of visual SLAM along with research areas such as dynamic scenes, adverse illumination, long-term map management, scalable dense mapping, uncertainty estimation, sensor calibration, semantic consistency, benchmarking standards, and deployment on resource-limited platforms. Continuous evolution of visual SLAM algorithms helps better perception and understandability of the environment, making them more accurate and robust.*

Keywords *- Backend, Comparative Study, Challenges, Frontend, Visual SLAM.*

1. Introduction

Autonomous systems, while building the model of their surroundings, must also know their location in that surrounding. This phenomenon is Simultaneous Localization and Mapping (SLAM). In Visual SLAM (or V-SLAM), the primary sensors are one or more cameras, which may or may not be supported by depth, event, inertial or any other complementary sensors. The problem of SLAM is not trivial because localization depends on a map of the surroundings that itself depends on the camera pose still to be evaluated. During practical deployment, the circular dependency of SLAM is complicated by the changes in illumination, surfaces with low texture, motion-blur, presence of dynamic objects, distortion of rolling-shutter, scale ambiguity in monocular vision, limited computational resources, and long-term operations in a complex environment.

Visual SLAM has become the fundamental perception technology for autonomous vehicles, unmanned aerial vehicles, mobile robots, augmented and virtual reality, agriculture, health care robotics, infrastructure inspection, industrial automation and underwater exploration. Cameras are preferred because they are compact, lightweight, inexpensive, information-rich and energy efficient. Images preserve colour, texture, object appearance and scene context, which is not possible in range-only sensors. However, this richness makes the localization problem sensitive to camera motion and scene appearance. Visual SLAM systems must be robust to solve several tasks, such as estimation of camera motion, choice of stable visual evidence, building and updating the map, detecting loop closure, correcting the drift that gets accumulated, to get computational feasible in real time and to reject the outliers.

The appropriate literature review must recognize the broader literature from which it evolved. Smith et al. [1, 2] through their seminal work on the estimation of spatial uncertainty, laid the foundation of SLAM. Durrant-Whyte et

al.[3, 4], later coupled the estimation problem and provided a mathematical link between localization and map building. Thurn et al. [5] provided a probabilistic framework for SLAM which involved Bayesian filtering, observation models, motion models and data association. Particle filter approaches such as FastSLAM [6], factor graphs [7], and Graph-SLAM [8] shaped estimation that is still used by visual SLAM. Uncertainity, inference, and optimization principles provided by this literature are the tools on which the front-end and backend of visual SLAM are built.

The classical visual SLAM pipeline has two stages: the frontend and the backend. The frontend converts the input image into geometric or learned-based cues, which include feature tracks, optical flow, photometric residuals, semantic masks, object landmarks and dense scene representations. The backend then estimates the appropriate camera trajectory and map by filtering, smoothing, Bundle Adjustment (BA), pose graph optimization or learned optimization modules. The modular view explains how the geometric methods, such as MonoSLAM [9], PTAM [10], ORB-SLAM [11], ORB-SLAM2 [12], ORB-SLAM3 [13], and DSO [14] can achieve real-time performance and accuracy. It also provides ways to understand new approaches that introduce learned features, dense bundle adjustment, neural matching, semantic mapping, differentiable SLAM, neural implicit scene representation. [15-20].

The growth of visual SLAM can be viewed as a gradual widening of the map concept. Early systems represented the environment as sparse landmarks that were sufficient for localization. In due course of time, the systems improved their scalability and accuracy through keyframe selection, loop closure, bundle adjustment, re-localization, and graph optimization. Systems with RGB-D or stereo were able to introduce metric dense or semi-dense maps, while visual-inertial systems were able to improve scale observability and robustness under rapid motion [21-23].

More recently, object-aware and semantic systems have moved beyond geometry-only maps by augmenting object labels, instant information, meshes, scene graphs, and topological relations to the estimated environment [24-29]. Neural implicit and differentiable formations, including iMAP, NICE-SLAM, DROID SLAM, gradSLAM, and related methods, have changed SLAM systems by optimizing camera pose and scene representation in such a way that they can now support dense reconstruction, novel view synthesis and end-to-end trainable components, [15, 18-20].

In spite of this progress in visual SLAM, important gaps are still present. Many good-performing visual SLAM systems are built assuming the observed world is static in nature, well-textured and observable. These assumptions limit the performance of SLAM systems in hospitals, homes, roads, warehouses, agricultural fields, underwater scenes, construction sites and crowded places. Dynamic objects often corrupt the correspondences between features and map points; low light and blurriness can break the tracking process; seasonal and structural changes can reduce the reliability of place recognition; large scaled operation can create a memory shortage and increase computational requirements.

Learning-based systems, in some cases, improve robustness, but the systems should be trained and have limited cross-domain generalization. These systems also have high GPU demand and reduced interpretability. Recent work on semantic SLAM, Neural implicit SLAM, real-time learned SLAM, structural scene abstraction shows clear progress but the literature is grouped across geometry based, semantic, object level, learning based, neural rendering and sensor fusion communities [30-37].

This review is motivated by that fragmentation. Prevailing surveys provide valuable treatments of visual SLAM methods [38], traditional to semantic developments [39], the broader robust perception view of SLAM [40], semantic SLAM [32], and specific subproblems such as loop closures [41]. However, there is still a need for a unified review that connects the classical geometric-based systems to recent learning-based systems also provide choice of algorithms depending upon the operating scenarios. Researchers and practitioners need a consolidated explanation of feature-based, direct, visual inertial, semantic, differentiable, object-level, neural implicit approaches that will be appropriate for a particular environment, along with its strengths and limitations and the challenges that will remain open for scalable, long-term and trusted autonomy.

Therefore, the objective of this article is to provide a comprehensive and structured review of visual SLAM from geometric to learning-based systems. The review spans from frontend architecture, which includes feature-based, direct, and semi-direct learned features, deep optical flow, semantic and object-aware approaches. It also surveys backend estimation, including filtering, factor graph optimization, bundle adjustment, outlier rejection, loop closure, visual-inertial fusion, and learning enhanced optimization.

Apart from this, it discusses how modern maps are represented, which includes sparse landmarks, dense and semi-dense geometry, object-level maps, semantic maps, scene graphs, differentiable maps, and neural representations.

The article further compares benchmark algorithms with respect to robustness, accuracy, sensing assumptions, computational requirements, map types, and suitability for outdoor, indoor, and dynamic environments. Finally, it provides open challenges and future directions that are related to scalability, dynamic scenes, semantic consistency, lifelong operation, real-time deployment, uncertainty quantification, and verifiable learning-based perception.

The literature in this review was selected using explicit inclusion and extrusion criteria that has to meet one or more of the following criterion: (i) propose a complete visual, monocular, stereo, RGB-D or visual inertial SLAM system; (ii) it should introduce front-end, back-end, mapping, loop closure, semantic, object level, differentiable or neural based method having direct relevance to visual SLAM; (iii) present prominent benchmark, comparative evaluation and open source library; (iv) provide a survey or review article that should help in organizing the field; or (v) present a seminal visual SLAM concept such as probabilistic uncertainty representation, Bayesian filtering, particle filter SLAM, smoothing, factor graphs, graph optimization, multiple-view geometry, or bundle adjustment. Priority was offered to peer-reviewed articles and conference publications, popularly cited seminal papers, recent preprints, and open source systems, from 2024 to 2026, when addressing topics that were not completely represented in the journal. Articles with pure visual odometry were excluded unless they impacted the front end or extended into visual SLAM systems. Works that were purely based on LiDAR SLAM, radar SLAM, wireless SLAM or localization without mapping were excluded unless useful in the context of sensor fusion.

The literature search was carried out across IEEE Xplore, MDPI, ScienceDirect, Google Scholar, SpringerLink, ACM Digital Library, arXiv, and major conference proceedings in robotics and computer vision, including ICRA, IROS, RSS, CVPR, ICCV, ECCV, NeurIPS, and IEEE Transactions on Robotics. The main keyword groups were: "visual SLAM", "VSLAM", "visual-inertial SLAM", "monocular SLAM", "RGB-D SLAM", "stereo SLAM", "semantic SLAM", "object-level SLAM", "dynamic visual SLAM", "deep visual SLAM", "learning-based SLAM", "differentiable SLAM", "neural implicit SLAM", "NeRF SLAM", "loop closure detection", "bundle adjustment", "factor graph SLAM", "GraphSLAM", "FastSLAM", "probabilistic robotics", and "SLAM survey". Boolean combinations such as "visual SLAM AND deep learning", "visual SLAM AND dynamic environments", "semantic SLAM AND review", and "neural implicit AND SLAM" were used to refine the search. The articles were searched from 2015 to June 2026 to cover the transition of visual SLAM from mature open source geometric systems, semantic, differentiable and neural SLAM. To establish the essential concepts, works on probabilistic SLAM, spatial uncertainty representation, particle filter SLAM, EKF based monocular SLAM, PTAM, bundle adjustment, factor graphs, graph optimization, and multiple view geometry [1-10, 42-47], and were also selected. The articles were then thematically organized so that the reader has a clear understanding of their progression from visual measurements to pose estimation, map representation, comparison of the systems and their future research requirements.

The subsequent sections of the article unfold as follows. Section 2 deals with the visual SLAM frontend, moving from geometric feature-based and direct methods to learned features, deep correspondence estimation, semantic segmentation, object-aware processing, and differentiable rendering. Section 3 explains the back-end that includes filtering, nonlinear optimization, pose graphs, loop closure, outlier handling, sensor fusion, and learning-enhanced estimation. Section 4 discusses the key trends that are shaping the modern visual SLAM systems. Section 5 provides a comparative analysis of different visual SLAM algorithms working under different environmental conditions and computational demands. Section 6 identifies major challenges that affect the implementation and unresolved research problems. Section 7 provides a summary of prominent open-source libraries and practical software ecosystems. Section 8 concludes the review and also gives future directions for scalable, adaptive, semantic and trustworthy visual SLAM.

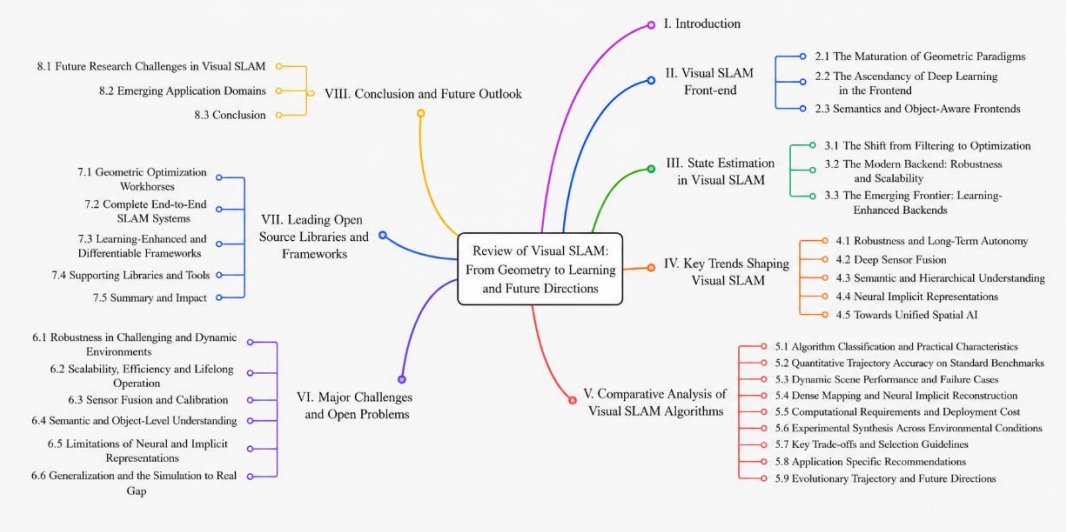

Fig. 1 Structure of the article

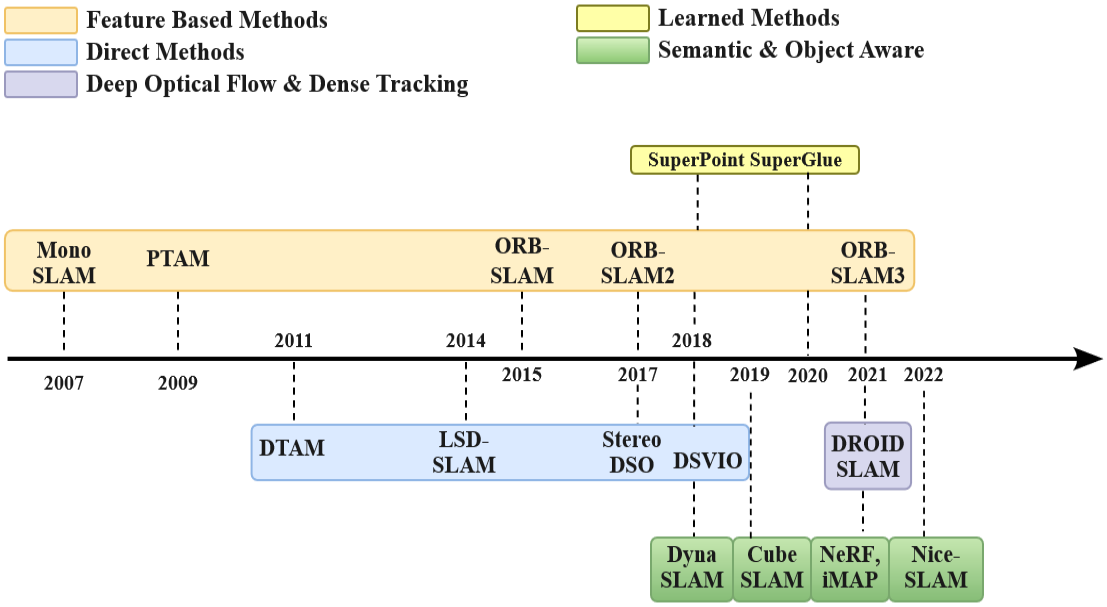

Fig. 2 Evolution of Visual SLAM Front-End Architecture

2. Visual SLAM Front-end

The front-end serves as the perceptual interface of a SLAM system. Its primary function is to process images sequentially to estimate ego-motion and locate the environmental landmarks on the map. The evolution of front-end architecture shows a continuous improvement to make SLAM more robust, accurate, and efficient.

2.1. The Maturation of Geometric Paradigms

The growth of front-end in visual SLAM methods, represented in Figure 2, shows the development of its approaches from time to time. Geometric methods, which depend on mathematical models of image formation and multi-view geometry, are categorized into feature-based and direct approach methods.

2.1.1. Feature-based Methods

In these methods, distinctive points such as corners, edges, and blobs called keypoints are detected from every image frame and are assigned mathematical descriptors. Then these descriptors are matched across different keyframes. Geometric relationships estimate the motion of the camera and help to build the 3-D map of its environment.

ORB-SLAM series [11-13] is a leading example of this approach. All of these architectures use ORB features [48] that maintain a balanced computational efficiency and geometric robustness, making visual SLAM feasible on the hardware of the system.

ORB-SLAM series simultaneously processes three threads, such as parallel mapping, tracking, and loop closing, independently, to make a robust and complete system.

The epipolar geometry constraint provides a mathematical framework to visual SLAM for the estimation of camera motion. Given a pair of images, the relationship between the corresponding image points p_1 and p_2 (in normalized coordinates) expressed as Equation 1 is given by [46]:

$$p_2^T F p_1 \tag{1}$$

where F is the fundamental matrix. For calibrated cameras, F simplifies to the essential matrix E as shown by Equation 2

$$p_2^T E p_1 \tag{2}$$

The essential matrix is then decomposed into the relative rotational matrix R and skew-symmetric translational matrix t between the two views of the camera: $E = [t]_x R$. This is typically solved using the five-point algorithm [49] within a robust RANSAC [50] framework for robustness.

For pose estimation from 3D-2D correspondences (Perspective-n-Point or PnP), the goal is to find the camera pose (R, t) that minimizes the reprojection error [46] written as Equation 3:

$$\min_{R,t} \sum_i \|u_i - \pi(RX_i + t)\|^2 \tag{3}$$

where X_i: a 3D point, u_i: it's measured 2D projection, and π is the projection function of the camera.

Recent advances focus on improving robustness under challenging conditions. Geometric Correspondence Network

(GCN) [51] which combines the Convolution Neural Network(CNN) with the Recurrent Neural Network(RNN) and trains them together. This helps it to detect keypoints and generate descriptors, and shows better motion estimation than hand-crafted methods. GCNv2 [52] built on GCN, had improved computational efficiency with accuracy comparable to the parent method.

2.1.2. Direct Methods

The direct approach method focuses on the raw pixel intensities of the entire image. The difference between the pixel intensities on the consecutive images, known as the photometric error, is minimized. The solution of photometric error minimization provides the estimated camera pose(motion).

Direct Sparse Odometry (DSO) [14] has a unified framework that combines "sparse", where it carefully selects a sparse set of uniformly distributed pixels and "direct", where the photometric error is directly minimized.

The photometric error [53] for a point p in the reference frame (I_i) as observed in the target frame (I_j) is defined as shown in Equation 4:

$$E_{qj} = \sum_{q \in N_q} w_q \left\| \left(I_j[\mathrm{q}'] - \beta_j\right) - \frac{t_j e^{\alpha_j}}{t_i e^{\alpha_i}} \left(I_i[\mathrm{q}] - \beta_i\right) \right\|_\gamma \quad (4)$$

where:

- N_q = small patch around q
- $\mathrm{q}' = \pi(T_{ji} \cdot \pi^{-1}(\mathrm{q}, d_\mathrm{q}))$ is the projected point position
- $T_{ji} \in SE(3)$, transformation from i^{th} to j^{th} frame
- α_i, α_j are affine brightness transfer parameters
- w_q is a gradient-dependent weighting
- $\|\cdot\|_\gamma$ represents the Huber norm
- $\beta_i, \beta_{j,}$ are camera bias parameters
- t_i, t_j are exposure times

The complete energy function [54] minimized in DSO is provided as Equation 5:

$$E = \sum_{i \in F} \sum_{\mathbf{q} \in \mathrm{Q}_i} \sum_{j \in obs(\mathbf{q})} E_{\mathbf{q}j} + \omega \left(w_\mathbf{q} \|\alpha_i - \alpha_j\|^2 + w_\mathbf{q} \|\beta_i - \beta_j\|^2 \right) \quad (5)$$

where F: frame set, Q_i: collection of points in i^{th} frame, and α is a weighting factor.

Extensions to DSO have addressed its limitations. Stereo DSO [55] incorporates stereo imagery to establish scale and improve robustness. Direct Sparse Visual-Inertial Odometry [56] tightly couples IMU measurements with the direct visual formulation.

2.2. The Ascendancy of Deep Learning in the Frontend

Deep learning used as a part of the front-end of SLAM provides a data-driven formulation that offers more robustness than handcrafted algorithms, in conditions dealing with the variation of light.

2.2.1. Learned Features and Detectors

Key advancement took place when the self-supervised system was created, which can learn to detect and describe interest points on its own without the requirement of labelling the training data manually.

SuperPoint [16] is a fully convolutional neural network model that operates on images and computes keypoints at the pixel level along with their descriptors in a single pass. The model, when trained with Homographic Adaptation, detects a rich set of important points as compared with traditional corner detection algorithms or pre-adapted deep learning models. The SuperPoint architecture can be formulated as shown by Equation 6:

$$\mathcal{D}, \mathcal{S} = f_\theta(I) \quad (6)$$

where f_θ: network with parameters θ, I: input image, $\mathcal{D}$: a dense descriptor map, and $\mathcal{S}$: a score map indicating keypoint probabilities.

Its successor, SuperGlue [17], introduces an attentional graph neural network that performs context-aware feature matching. The matching is formulated as an optimal transport problem, solving for a partial assignment matrix Q that maximizes the total score as provided by Equation 7:

$$\max_Q \sum_{i=1}^{M} \sum_{j=1}^{N} \mathrm{S}_{i,j}\ \mathrm{Q}_{i,j} \quad (7)$$

subject to assignment constraints, where $\mathrm{S}_{i,j}$: similarity score between i^{th} and j^{th} descriptors of the first and the second image, respectively; *M*, *N:* the number of features in the first and second image, respectively.

2.2.2. Deep Optical Flow and Dense Tracking

Instead of relying on sparse features, some modern frontends use deep networks to compute dense or semi-dense correspondences, effectively "learning" the correspondence search. DROID-SLAM [15] is a SLAM architecture based on deep learning that uses a learned recurrent network to construct a dense pixel-level correspondence graph across all frames and iteratively refine camera poses and scene geometry through learned updates.

Optimization is performed using Gauss-Newton [57] iterations to minimize the cost function of the residual shown in Equation 8:

$$\Delta \mathrm{p} = -(\mathrm{J}^T \mathrm{W} \mathrm{J})^{-1} \mathrm{J}^T \mathrm{W} \mathrm{e} \quad (8)$$

Where $p \in \mathfrak{se}(3)$ is the camera pose, J : Jacobian of

residuals corresponding to the pose, W: weight matrix, and e: residual vector.

2.3. Semantics and Object-Aware Frontends

The next development is to move from pure geometry to semantic understanding, which enables the creation of meaningful, hierarchical maps and providing object level constraints.

2.3.1. Semantic Segmentation for Robust Tracking

The frontend can classify and filter dynamic objects by making use of semantic segmentation. Dynaslam [30] is capable of detecting dynamic objects and performs background inpainting.

It uses R-CNN [58] for detecting and masking dynamic objects, and then it is used for pose estimation. The segmentation model minimizes the cross-entropy [59] loss as provided by Equation 9:

$$L_{seg} = -\sum_{i=1}^{H}\sum_{j=1}^{W}\sum_{c=1}^{C} t_{i,j,c}\log\left(\hat{t}_{i,j,c}\right) \quad (9)$$

where $t_{i,j,c}$: label for ground truth and $\hat{t}_{i,j,c}$: predicted probability for class c at pixel (i, j).

2.3.2. Object-Level SLAM

In object-level SLAM, there is a deeper level of integration where it filters the dynamic objects and uses them as landmarks for pose estimation. CubeSLAM [60] formulates 3D object detection and SLAM as a joint optimization problem. For each object observation, it minimizes the error between the bounding box (in 2D) and the projection of the cuboid (in 3D) shown in Equation 10:

$$E_{object} = \sum_{k}\left\|\boldsymbol{b}_k - \mathcal{B}\left(\pi(\boldsymbol{T}_{cw}\cdot\boldsymbol{C}_w)\right)\right\|^2_{\Sigma_k} \quad (10)$$

where b_k: detected 2D bounding box, $\mathcal{B}$ extracts the bounding box from the projected cuboid corners, T_{cw}: camera pose, and C_w: 3D cuboid in world coordinates.

2.3.3. Differentiable Rendering for Dense Alignment

A groundbreaking trend is the integration of neural radiance fields (NeRF) [25] with the SLAM loop. Systems such as iMAP [19] and NICE-SLAM [20] use a neural implicit representation as the map. The core NeRF model that represents a scene with a continuous function which maps a 3D location '**x**' and viewing direction 'd' to a colour 'c' and density 'σ' as expressed in Equation 11:

$$(\boldsymbol{c}, \sigma) = f_\theta(\mathrm{x}, \mathrm{d}) \quad (11)$$

where, f_θ represents a multilayer perceptron (MLP) with parameters θ.

Rendering an image involves computing the integral along accumulated transmittance T(t) , colour (c(r), d)), density (σ(r)) as shown in Equation 12:

$$\hat{I}(\boldsymbol{r}) = \int_{t_n}^{t_f} T\,(t)\sigma\big(r(t)\big)c(r(t), d)\,dt \quad (12)$$

where,

- $\hat{I}(r)$ = rendered image,
- $T(t) = \exp\left(-\int_{t_n}^{t}\sigma\left(r(s)\right)ds\right)$
- camera ray: $r(t) = o + td$
- t_n, t_f are the near and far bound distances, respectively
- d: direction of the ray

The photometric error between the rendered and the observed images is minimized by Equation 13:

$$E_{nerf} = \sum_{r\in\mathcal{R}}\left\|\hat{I}(\boldsymbol{r}) - I(\boldsymbol{r})\right\|^2 \quad (13)$$

where $\mathcal{R}$: set of rays in the current frame and $C(\boldsymbol{r})$: observed colour. It is then used in SLAM.

3. State Estimation in Visual SLAM: The Reasoning Engine

The backend is the reasoning engine of a SLAM system, tasked with achieving global consistency by optimizing the estimated trajectory and map based on all accumulated sensor data and constraints.

3.1. The Shift from Filtering to Optimization

History has seen the evolution of SLAM back-ends from recursive methods to batch optimization methods, as shown in Figure 3.

3.1.1. The Filtering Era

Early visual SLAM systems were particularly based on recursive Bayesian estimators. The Extended Kalman Filter (EKF) was the prevalent approach, as illustrated by MonoSLAM [9]. In EKF [5], the state vector $\boldsymbol{x}_t$, contains the robot pose as well as landmark positions, and the state vector is then represented in the form of a Gaussian belief as provided by Equation 14:

$$bel(x_t) = \mathcal{N}(x_t; \mu_t, \Sigma_t) \quad (14)$$

The algorithm operates in two stages: the prediction stage and the update stage. The prediction stage is represented via Equations 15, 16:

$$\mu_t = \Phi(\mu_{t-1}, u_t) \quad (15)$$

$$\Sigma_t = \Phi_t\Sigma_{t-1}\Phi_t^T + R_t \quad (16)$$

Where Φ represents the motion model, u_t as the control input, Φ_t is the Jacobian for Φ, and R_t is the covariance for motion noise.

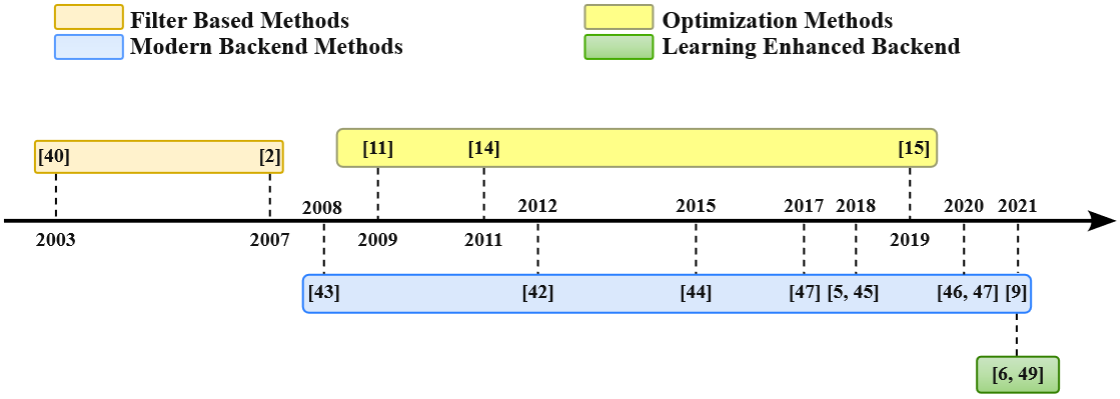

Fig. 3 Evolution of Visual SLAM Back-End Architecture

The update stage is represented as Equations 17, 18 and 19:

$$K_t = \Sigma_t H_t^T (H_t \Sigma_t H_t^T + Q_t)^{-1} \quad (17)$$

$$\mu_t = \mu_t + K_t\big(z_t - h(\mu_t)\big) \quad (18)$$

$$\Sigma_t = (I - K_t H_t)\Sigma_t \quad (19)$$

where h: the measurement model, H_t: Jacobian, z_t: the measurement, and Q_t: covariance of the measurement noise. The computational complexity of the EKF is of order two with landmarks given as $O(n^2)$. This limits the use of EKF for large-scale environments.

Particle filters offered an alternative. FastSLAM 2.0 [6] used a Rao-Blackwell particle filter. In these filters, each particle carries its own trajectory estimate and a set of independent EKFs for landmark estimation. Although more scalable than the full EKF, it suffered from particle depletion and was still less accurate than later optimization-based methods.

3.1.2. The Optimization Revolution

The pivotal insight, famously framed as "Why filter?" by Strasdat et al. [61], was that the SLAM problem is inherently sparse and is better solved as a large-scale batch optimization over a history of states.

Parallel Tracking and Mapping (PTAM) [10] played a crucial role during the development of visual SLAM. PTAM detached Tracking process, a part of the frontend and Mapping process, a part of the backend, are split into two separate threads that execute simultaneously. PTAM also introduced the concept of keyframes, using a sparse set of frames for mapping, which made computationally expensive Bundle Adjustment (BA) [43] feasible.

Factor Graphs [7] formalise the sparse structure for the BA optimization problem. SLAM represented by a bipartite graph, having the robot pose x_i, landmark position I_j as variable nodes and measurements z_k as factor nodes. The joint probability distribution given as Equation 20:

$$P(X, L|Z) \propto \prod_k P\left(z_k|x_{i_k}, l_{j_k}\right) \quad (20)$$

The SLAM problem then gets converted to Maximum A Posteriori (MAP), expressed as Equation 21:

$$X^*, L^* = \arg\max_{X,L} \prod_k P\left(z_k|x_{i_k}, l_{j_k}\right) \quad (21)$$

Assuming Gaussian noise, it becomes a non-linear least squares problem from Equation 22:

$$X^*, L^* = \arg\min_{X,L} \sum_k \left\| z_k - h_k\left(x_{i_k}, l_{j_k}\right)\right\|^2_{\Sigma_k} \quad (22)$$

Bundle Adjustment (BA) [43] applies this formulation to the visual SLAM problem, where the factors are typically the reprojection error given by Equation 23:

$$E_{BA} = \sum_i \sum_j \left\| p_{ij} - \pi\left(T_{iw} l_j^w\right)\right\|^2_{\Sigma_{ij}} \quad (23)$$

The efficiency of this approach comes from the sparse nature of the underlying matrices. The Hessian matrix $H = \mathrm{J}^T\mathrm{J}$ exhibits a distinctive block-structured form shown in Equation 24.

$$H = \begin{bmatrix} H_{XX} & H_{XL} \\ H_{XL}^T & H_{LL} \end{bmatrix} \quad (24)$$

Using the Schur complement, the landmark variables are marginalized as provided by Equation 25:

$$H_{XX}^{*} = H_{XX} - H_{XL}H_{LL}^{-1}H_{XL}^{T} \quad (25)$$

This reduces the optimization problem to only the pose variables by eliminating the landmarks, which reduces the computational time and makes the solution faster even for large-scale environments.

3.2. The Modern Backend: Robustness and Scalability

The focus of research on modern backends is to improve the robustness to perpetual aliasing and to enable lifelong operation.

3.2.1 Robust Loop Closure and Outlier Rejection

The entire map in SLAM may get corrupted due to incorrect loop closure. Recent back-ends employ advanced techniques for robustness.

Robust kernels prevent wrong or corrupt data from dominating the optimization process and thus improves stability, accuracy and robustness in SLAM systems. They replace squared errors with functions that give less weight to large residuals, as shown in Equation 26.

$$E_{robust} = \sum_k \rho\left(\left\|z_k - h_k\left(x_{i_k}, l_{j_k}\right)\right\|_{\Sigma_k}\right) \quad (26)$$

where $\rho(.)$ represents a cost function such as Huber or Cauchy.

Switchable constraints [62] introduce a switch variable s_k for each loop closure from Equation 27:

$$E_{switch} = \sum_k s_k \left\|\boldsymbol{z}_k - \boldsymbol{h}_k\left(\boldsymbol{x}_{i_k}, \boldsymbol{l}_{j_k}\right)\right\|_{\Sigma_k}^2 + \sum_k \phi\left(s_k\right) \quad (27)$$

where $\phi(s_k)$ is a prior that encourages s_k to be 1 (on).

3.2.2. Multi-Sensor Fusion

The idea of sliding window optimization was formulated by [63] where only the recent measurements are retained for state estimation, providing a balance between accuracy and computational time.

Visual-Inertial Odometry (VIO) systems like VINS-Mono [21, 22] applied sliding window optimization for around 10 keyframes. Direct Sparse Odometry [14], LiDAR-Inertial System [64] and Multi-sensor fusion system [13, 65], made use of sliding window optimization in their works.

Instead of integrating IMU measurements between each pair of camera frames, which would require reintegration when the initial state changes, IMU measurements collected between two keyframes are pre-integrated [23] as a single motion constraint shown in Equation 28:

$$\Delta R_{ij}, \Delta v_{ij}, \Delta p_{ij} = Preintegrate\left(\{\omega_k, a_k\}_{k=i}^{j-1}\right) \quad (28)$$

The IMU factor between two keyframes then becomes as shown in Equation 29:

$$r_j = \begin{bmatrix} r_{\Delta R_{ij}} \\ r_{\Delta v_{ij}} \\ r_{\Delta p_{ij}} \end{bmatrix} = \begin{bmatrix} \log\left(\Delta R_{ij}^T R_i^T R_j\right) \\ R_i^T\left(v_j - v_i - g\Delta t_{ij}\right) - \Delta v_{ij} \\ R_i^T\left(p_j - p_i - v_i\Delta t_{ij} - \frac{1}{2}g\Delta t_{ij}^2\right) - \Delta p_{ij} \end{bmatrix} \quad (29)$$

3.3. The Emerging Frontier: Learning-Enhanced Backends

Deep learning is beginning to influence the core optimization processes of the backend.

- Learned Uncertainty Estimation: Traditional SLAM assumes a fixed noise model. Learning can predict more accurate, data-dependent uncertainties. The measurement covariance Σ_k can be learned through a neural network, as shown in Equation 30:

$$\Sigma_k = f_\phi\left(I_i, I_j, p_i, p_j\right) \quad (30)$$

where f_ϕ is a neural network with parameters ϕ [66].

- Differentiable SLAM: In DROID-SLAM [15], the back-end optimization is not a separate module but is embedded within a deep network that uses a learned iterative update rule. The network learns to predict updates through Equation 31:

$$\Delta\xi, \Delta d = g_\psi(r, j) \quad (31)$$

where g_ψ is a network that takes residuals and Jacobians as input and predicts updates to poses ξ and depths d.

4. Key Trends Shaping the Visual SLAM

Several convergent trends are redefining the capabilities and ambitions of visual SLAM research, moving it from a geometric state-estimation module towards a core component of general spatial AI.

4.1. Robustness and Long-Term Autonomy

The primary challenge for real-world deployment is achieving robust performance over extended periods in ever-changing environments. This has led to research on multimap systems [13] that maintain a set of submaps $\{M_1, M_2, \ldots, M_N\}$ with associated pose graphs. Map merging is formulated as an optimization over the relative transformations between submaps shown in Equation 32:

$$\min_{\{T_{ij}\}} \sum_{i,j} \left\|\boldsymbol{T}_{ij} \ominus \hat{\boldsymbol{T}}_{ij}\right\|_{\Sigma_{ij}}^2 \quad (32)$$

where T_{ij}: relative transformation between submap i, j, and $\hat{T}_{ij}$: measured relative transformation.

Beyond multimap systems, handling dynamic environments remains crucial. Approaches like DynaSLAM [30] use semantic segmentation to identify and exclude

dynamic objects, while more advanced methods like Mid-Fusion [26] represent dynamic objects as separate instances on the map. The energy function as provided by Equation **Error! Reference source not found.** for dynamic SLAM can be extended to include object motion models:

$$E_{dynamic} = E_{static} + \sum_{k=1}^{K} \left\| z_k^{obj} - h_k^{obj}(T_{cw}, O_k) \right\|_{\Sigma_k^{obj}}^2 \quad (33)$$

where O_k represents the state for the k-th dynamic object.

4.2. Deep Sensor Fusion

Pure vision is fragile. The trend is toward deeply integrating complementary sensors. The joint visual-inertial optimization problem combines visual reprojection errors with IMU preintegration factors expressed as Equation 34:

$$E_{vio} = \sum_{i,j} \left\| \boldsymbol{r}_{v_{ij}} \right\|_{\Sigma_{v_{ij}}}^2 + \sum_{k,l} \left\| \boldsymbol{r}_{j_{kl}} \right\|_{\Sigma_{j_{kl}}}^2 + \sum_{m,n} \left\| \boldsymbol{r}_{\mathcal{L}_{mn}} \right\|_{\Sigma_{\mathcal{L}_{mn}}}^2 \quad (34)$$

where r_v, r_j, and $r_{\mathcal{L}}$ are visual, inertial, and (if applicable) LiDAR factors, respectively.

Recent approaches also integrate event cameras [67], which provide asynchronous brightness changes. The fusion of event data with standard frames formulated by Equation 35:

$$E_{event} = \sum_{e \in \mathcal{E}} \| L(\boldsymbol{x}_e, t_e) - L(\boldsymbol{x}_e, t_{e-1}) - C \cdot \Delta L_e \|^2 \quad (35)$$

where $L(x, t)$: logarithmic intensity in pixel $\boldsymbol{x}$ at time t, C: contrast threshold, and ΔL_e: polarity of event.

4.3. Semantic and Hierarchical Understanding

The push for more intelligent perception is driving the integration of semantics. Kimera [27] builds a 3D semantic mesh by combining semantic labels from 2D segmentation with geometric information. The semantic labelling can be formulated as a Conditional Random Field (CRF) over the mesh vertices, shown as Equation 36:

$$P(I|V) \propto \exp\left(-\sum_{m} \phi_u(l_m, V) - \sum_{m,n} \phi_p(l_m, l_n, V) \right) \quad (36)$$

where I are semantic labels, V are vertex positions, ϕ_u are unary potentials from the 2D segmentation, and ϕ_p are pairwise potentials that encourage label smoothness. Object-level SLAM systems like CubeSLAM [60] and SO-SLAM [28] extend this by incorporating object shape priors and physical constraints. The joint optimization includes object-level factors provided by Equation 37:

$$E_{object} = \sum_k \left(\left\| b_k - \mathcal{B}\left(\pi(T_{cw} C_w)\right) \right\|^2 + \lambda_{shape} \left\| C_w - C_{prior} \right\|^2 \right) \quad (37)$$

where λ_{shape} weights the shape prior term.

4.4. Neural Implicit Representations

This is arguably the most disruptive trend in map representation. The neural implicit map in iMAP [19] and NICE-SLAM [20] is pre-trained, and the loss of photometric rendering is minimized by Equation 38:

$$\mathcal{L} = \sum_{\boldsymbol{r} \in \mathcal{R}} \left\| \hat{I}(\boldsymbol{r}) - I(\boldsymbol{r}) \right\|^2 + w_{depth} \left\| \hat{D}(\boldsymbol{r}) - D(\boldsymbol{r}) \right\|^2 + \lambda_{reg} \mathcal{R}(\theta) \quad (38)$$

where $\hat{D}(r)$: rendered depth, $D(r)$: measured depth (for RGB-D), and $\mathcal{R}(\theta)$: regularization term on the network parameters. To address scalability limitations, hierarchical representations like those in NICE-SLAM employ multiresolution feature grids given by Equation 39:

$$f_\theta(x, d) = MLP(\phi_1(x), \phi_2(x), \ldots, \phi_L(x), d) \quad (39)$$

Where, $\phi_i(\boldsymbol{x})$: features of different resolution levels. Recent advances [68, 69] further improve efficiency through hash encodings and tensor factorizations expressed by Equation 40:

$$\phi(\mathrm{x}) = \mathrm{HashGrid}(\mathrm{x}) \quad \text{or}$$

$$\phi(\mathrm{x}) = \prod_{d=1}^{D} \mathcal{T}_d(\mathrm{x}_d) \quad (40)$$

which enables real-time performance and large-scale reconstruction.

4.5. Towards Unified Spatial AI

Recent trends are moving towards unified spatial AI systems that combine geometric understanding, semantic understanding and physical awareness. The advanced version of visual SLAM can be expressed as a hierarchical generative model provided as Equation 41:

$$P(X_{1:T}, L, O_{1:K}, S \mid Z_{1:T}) \propto \underbrace{P(Z_{1:T} \mid X_{1:T}, L, O_{1:K}, S)}_{\text{Measurement model}} \underbrace{P(O_{1:K} \mid S)}_{\text{Object prior}} \underbrace{P(S)}_{\text{Semantic prior}} \quad (41)$$

where $X_{1:T}$: robot trajectories, L: geometric landmarks, $O_{1:K}$: object instances, and S: semantic scene structure.

The optimization objective becomes multimodal (as per Equation 42, which integrates geometric constraints, semantic knowledge, actual physical constraints, and temporal consistency in a single architecture, incorporating geometric consistency, semantic coherence, physical plausibility, and temporal smoothness in a single framework.

$$E_{unified} = E_{geometry} + \lambda_{semantic}E_{semantic} + \lambda_{physical}E_{physical} + \lambda_{temporal}E_{temporal} \quad (42)$$

5. Comparative Analysis of Visual SLAM Algorithms

In the previous section, the article described visual SLAM from the perspective of frontend design, backend estimation, semantic mapping, and learning based representative. This section compares SLAM systems across these dimensions in an application-oriented way.

5.1. Algorithm Classification and Practical Characteristics

Table 1 provides a comparison of representative algorithms, mainly feature-based, direct, semi-direct, visual-inertial, semantic, learned and neural implicit. It highlights three broad patterns. First, classical geometric methods are preferable during real-time operation, explainable residuals and modest hardware. Second, the effect of fast motion and visual dropout is reduced by visual-inertial methods, but they introduce calibration and initialization failure modes. Third, neural implicit and learned methods provide strong dense tracking and reconstruction in many benchmarks, but at the cost of GPU dependence, reduced interpretability and performs weak on domains that are not trained or evaluated.

5.2. Quantitative Trajectory Accuracy on Standard Benchmarks

Table 2 gives a summary of the results where methods based on classical, dense and learned approaches were evaluated on different datasets such as TartanAir, EuRoC and TUM-RGBD.

Table 1. Standardized qualitative comparison of representative visual SLAM systems

Method	Paradigm	Sensors used in the reported system	Map representation	Main strengths	Main limitations and typical failure cases	Suitable use cases
ORB-SLAM3 [13]	Feature-based visual/visual-inertial SLAM	Monocular, stereo, RGB-D, IMU	Sparse keyframes and landmarks; multi-map Atlas	Mature full SLAM; loop closure; re-localization; visual-inertial support; efficient CPU execution	Needs sufficient repeatable features; can fail in severe blur, low texture, strong illumination change, or highly dynamic scenes	AR/VR, mobile robotics, indoor/outdoor navigation with moderate texture
DSO [14]	Direct sparse visual odometry	Monocular	Sparse photometric points	Uses intensity information directly; accurate in many textured scenes; avoids descriptor matching	VO rather than complete SLAM; no native loop closure or re-localization; sensitive to photometric calibration, exposure changes, rolling shutter, and dynamic objects	Research on direct VO; locally accurate motion estimation
SVO 2.0 [70]	Semi-direct VO	Monocular and multi-camera	Sparse landmarks	Very fast; suitable for embedded and high-speed platforms; combines direct tracking with feature-based mapping.	Limited global consistency without loop closure; sparse map; less suitable when dense reconstruction or semantic	UAVs, embedded platforms, low-latency tracking

					understanding is needed	
VINS-Mono [22]	Visual-inertial odometry/SLAM	Monocular camera + IMU	Sparse visual-inertial keyframes	Robust under short visual degradation; online temporal calibration; loop closure and re-localization support	Requires correct initialization and calibration; feature tracking can still fail in texture-poor or highly dynamic scenes.	Drones, handheld devices, agile motion, mobile robots
DynaSLAM [30]	Dynamic-scene semantic SLAM	Monocular, stereo, RGB-D	Sparse static map with dynamic object masking; optional inpainting	Improves tracking and mapping in scenes containing known moving objects	Depends on semantic segmentation and multi-view geometry; computationally heavier; may miss unknown dynamic objects or non-rigid motion	Indoor service robots, human-populated environments
Kimera [24, 29]	Metric-semantic SLAM	Stereo/RGB-D + IMU variants	Metric mesh, semantic mesh, scene graph	Links VIO, meshing, semantic labels, and 3D dynamic scene graphs	Higher system complexity; semantic quality depends on segmentation; object reasoning is not free in real time	Spatial AI, robot planning, human-robot interaction
DROID-SLAM [15]	Learned dense visual SLAM	Monocular, stereo, RGB-D	Dense depth and trajectory optimized by learned recurrent updates	Very strong cross-dataset accuracy; robust dense correspondence; works across multiple camera settings	Requires GPU; less interpretable than geometric optimization; generalization still depends on training distribution	Accuracy-focused benchmarking, dense tracking, and research platforms
iMAP [19]	Neural implicit RGB-D SLAM	RGB-D	Continuous implicit scene representation	Joint tracking and dense mapping; compact neural scene model	Room scale focus; slow optimization; GPU memory demand; vulnerable to dynamic scenes and blur	Small-scale dense mapping, novel view-oriented research
NICE-SLAM [20]	Neural implicit RGB-D SLAM	RGB-D	Hierarchical neural implicit grid	Improves scalability over a single MLP representation; dense reconstruction.	Still needs a GPU and RGB-D input; less suitable for large outdoor spaces or strict real-time embedded deployment.	Indoor dense reconstruction and neural mapping

On the TartanAir hard-split dataset, DROID-SLAM shows a 87.5% and 95.2% reduction in ATE (RMSE) on TartanVO and DeepV2D approaches, respectively. On the TartanAir competition test set, the ATE value of DROID-SLAM is 62.1% lower than SuperGlue + SuperPoint + COLMAP approaches. On the EuRoC dataset, the reported mean ATE of DROID-SLAM reported a reduction of 82.5% relative to DSM and about 96.3% relative to DSO approaches. On the TUM-RGBD dataset, the DROID-SLAM also shows a reduction of 89.8% and 83.6% relative to DeepV2D and DeepFactors, respectively, in its ATE values. ORB-SLAM3 [9] is a competitive SLAM system when loop closure, re-localization and multi-map operations are important, but it still did not complete all TUM-RGBD sequences, and when tested on EuRoC, it reported a higher mean ATE than DROID-SLAM [6]. This shows that sparse geometric pipelines and learned dense optimizers solve different operating problems.

Table 2. Reported trajectory accuracy comparison from DROID-SLAM experiments [15]

Dataset	Metric	Classical baseline	Learned baseline 1	Learned baseline 2	DROID-SLAM [6]	Critical interpretation
TartanAir hard monocular split	ATE RMSE	ORB-SLAM: failed on two reported sequences; no average reported	DeepV2D: 5.03	TartanVO: 1.92	0.24	DROID-SLAM gives much lower error on the synthetic hard split, but the synthetic-to-real transfer must still be checked.
TartanAir SLAM competition test set	Normalized relative pose error	SuperGlue + SuperPoint + COLMAP: 0.340	OV^{2}SLAM: 0.510	VOLDOR + COLMAP: 0.440	0.129	DROID-SLAM reports 62% lower monocular benchmark error than the best listed submission and runs faster than COLMAP-based pipelines.
EuRoC MAV	ATE RMSE	DSO: 0.601; ORB-SLAM3 has one failure and no full average	DeepV2D: 1.298	DSM: 0.126	0.022	DROID-SLAM reports 2.2 cm average ATE and 82% lower error among methods with zero failures; ORB-SLAM3 remains strong on successful sequences.
TUM-RGBD freiburg1	ATE RMSE	ORB-SLAM3 failed on 5 of 9 reported sequences	DeepV2D: 0.375	DeepFactors: 0.233	0.038	DROID-SLAM succeeds on all reported sequences and gives a much lower mean ATE. Failure count is as important as average error.

5.3 Dynamic Scene Performance and Failure Cases

Most of the visual SLAM systems assume a static environment for its working. DynaSLAM addresses this problem by combining ORB-SlAM2 with multi-view geometry and Mask R-CNN to remove dynamic regions [30, 58]. Mean ATE values of Dyna-SLAM were compared with ORB-SLAM3 when evaluated on two group sequences of the TUM-RGBD dataset, as shown in Table 3.

In highly dynamic conditions, as shown in fr3/walking sequences, the mean ATE values of the Dyna-SLAM approach were lower by 94.8%. This signifies its robustness by preventing the moving objects from corrupting the map. In a static background, as in the case of fr3/sitting sequences, the mean ATE error was 10.3% higher than that of ORB-SLAM3. This shows that Dyna-SLAM works better than ORB-SLAM3 in dynamic conditions but has limited performance in static conditions. Dyna-SLAM also depend upon known object classes from its model; unknown moving objects, transparent objects, deformable surfaces, reflections and foliage become hard to mask reliably.

Recent systems like VAR-SLAM target this limitation. On challenging dynamic sequences, VAR-SLAM [34] shows 25% lower ATE error than NGD-SLAM [71] while maintaining an average of 27 fps. This shows that dynamic scene-based SLAM approaches are moving from fixed semantic removal towards adaptive robust estimation, but it confirms that dynamic robustness can be measured together with runtime.

Table 3. Reported DynaSLAM performance on TUM-RGBD dynamic sequences [30]

Sequence group	ORB-SLAM2 mean ATE(m)	DynaSLAM mean ATE(m)	Relative change	Interpretation
fr3/sitting sequences	0.0145 m	0.0160 m	10.3% higher error	When the scene is only mildly dynamic, semantic masking gives little benefit and can slightly degrade accuracy because useful image regions may be removed.
fr3/walking sequences	0.3905 m	0.0203 m	94.8% lower error	When dynamic objects dominate the foreground, removing moving regions greatly improves tracking and map consistency.

5.4. Dense Mapping and Neural Implicit Reconstruction

Neural implicit SLAM methods are able to optimize a continuous scene representation and can produce dense reconstruction. Table 4 summarizes a few neural slam approaches, their representation, the datasets on which they were validated, and their limitations. ORB-SLAM3 are efficient in camera pose estimation and loop closure, but provides sparse landmark maps. Such maps are not sufficient surface-level reasoning. Neural implicit methods provide a rich map, but with the limitation of deployment on embedded robots, has slow speed and are more memory intensive. The selection of a proper neural implicit SLAM system depends on the map utility, hardware cost and application need.

Table 4. Neural SLAM approaches

Method	Representation	Reported benchmark setting	Reported quantitative evidence	Main limitation
iMAP [19]	Single neural implicit MLP	Room-scale RGB-D scenes	Demonstrates joint online tracking and dense implicit mapping	Limited scalability because one network must represent the scene; it can forget earlier regions during online updates
NICE-SLAM [20]	Hierarchical neural implicit grid	Replica, ScanNet, and other indoor RGB-D evaluations	Reports efficient, scalable, and robust dense mapping than earlier neural implicit SLAM	Requires GPU and depth input; still mainly demonstrated on indoor-scale scenes
UncLe-SLAM [72]	Dense neural SLAM with learned uncertainty	7-Scenes, TUM-RGBD, and Replica	Reports 38% lower ATE on 7-Scenes, 27% lower ATE on TUM-RGBD, and 11% F1-score improvement on Replica compared with recent neural implicit baselines	Adds uncertainty, learning complexity and remains within the dense neural SLAM hardware regime
DROID-SLAM [15]	Learned dense depth and recurrent update operator	TartanAir, EuRoC, TUM-RGBD trajectory evaluation	Reports 0.24 ATE on the TartanAir hard monocular split, 0.022 on EuRoC, and 0.038 on TUM-RGBD in the comparison summarized in Table 2	A dense map is not the same as a globally consistent semantic or neural scene model.

5.5. Computational Requirements and Deployment Cost

Table 5 provides the qualitative summary of various visual SLAM algorithms depending upon their resources used, real-time suitability and practical deployment. The comparison shows that the system, which shows high accuracy and not always easily deployable. ORB-SLAM3, SVO and VINS-Mono are preferred as they can run without a GPU but are exposed to interpretable geometric errors. DROID-SLAM and neural implicit methods are more rich in dense map reconstruction, but at the cost of GPU requirement, the model is difficult to generalize and has high power consumption at runtime. Dyna-SLAM and Kimera offer semantic understanding but add latency and segmentation error.

Table 5. Standardized deployment-oriented resource comparison

Method	Main computational load	GPU dependency	Memory pressure	Real-time suitability	Practical deployment concern
ORB-SLAM3 [13]	Feature extraction, matching, local BA, loop closure	No GPU required	Low to moderate	Strong on CPU	Tracking can fail if features are insufficient or corrupted by blur/dynamics.
DSO [14]	Photometric optimization over selected pixels	No GPU required	Moderate	Real-time CPU operation reported	Requires careful photometric calibration; lacks full SLAM functions unless extended
SVO 2.0 [70]	Fast direct tracking and sparse mapping	No GPU required	Low	Very strong for embedded/high-speed use	Accuracy and global consistency can be lower than heavier SLAM systems
VINS-Mono [22]	Feature tracking, IMU pre-integration, nonlinear optimization	No GPU required	Moderate	Real-time CPU operation reported	Initialization, camera-IMU calibration, and time synchronization are critical.
DynaSLAM [30]	ORB-SLAM2 plus semantic segmentation and dynamic masking	GPU is strongly beneficial for segmentation	Moderate to high	Slower than base ORB-SLAM2 when segmentation is active	Dynamic robustness is obtained at the cost of segmentation latency
Kimera [24, 29]	VIO, meshing, semantic fusion, scene graph construction	Recommended for semantic modules	High	Real-time components reported, but the full semantic stack is heavier	System integration and semantic segmentation quality affect reliability
DROID-SLAM [15]	Dense correlation, recurrent updates, learned optimization	Required in practical use	High	Real-time or near real-time depends on the GPU	Strong accuracy but higher energy demand and lower interpretability
iMAP [19]	Online neural optimization	Required	High	Limited to strict real-time deployment	Room-scale focus and optimization cost limit broad robotic deployment
NICE-SLAM [20]	Neural implicit grid optimization	Required	High	Better scalability than iMAP, but still GPU-bound	Indoor RGB-D focus; not a lightweight localization method

5.6. Experimental Synthesis across Environmental Conditions

Table 6 qualitatively summarizes visual SLAM systems across different operating conditions. Feature-based systems show dominance where texture and loop closures are available. Direct and learned methods improves image evidence, but direct methods fail when photometric assumptions are violated, whereas learned methods fail when there is a shift in domain. Semantic systems are robust when dynamic objects can be detected, masked or modelled. Neural implicit methods are best for dense mapping systems rather than replacements for lightweight SLAM.

Table 6. Evidence-linked synthesis of robustness across operating conditions

Method	Texture-rich scenes	Low-texture scenes	Illumination change	Motion blur / fast motion	Dynamic objects	Evidence-supported interpretation
ORB-SLAM3 [13]	Strong	Weak to moderate	Moderate	Weak to moderate	Weak	Sparse features and loop closure work well when repeatable features exist, but the method remains vulnerable when feature extraction or matching collapses.
DSO [14]	Strong	Moderate to strong	Weak to moderate	Moderate	Weak	Direct photometric alignment can use more image information than sparse descriptors, but it assumes brightness consistency and has no native loop closure.
SVO 2.0 [70]	Strong	Moderate	Moderate	Strong	Weak	The semi-direct design supports high-speed operation, but sparse mapping and limited global correction remain concerns.
VINS-Mono [22]	Strong	Moderate	Moderate	Strong	Weak to moderate	IMU constraints help during rapid motion or short visual degradation, but moving features and calibration errors can still cause drift.
DynaSLAM [30]	Strong	Weak to moderate	Moderate	Weak to moderate	Strong for known object classes	Table 3 shows a large improvement in walking sequences, but little gain in mildly dynamic scenes and dependence on segmentation.
DROID-SLAM [15]	Strong	Strong	Strong in reported benchmarks	Strong	Moderate to strong	Table 2 shows strong cross-benchmark trajectory results, but generalization and GPU dependence remain practical limitations.
iMAP [19] / NICE-SLAM [20]	Strong indoors with RGB-D	Moderate to strong with depth	Moderate	Weak to moderate	Weak	Dense neural maps are useful indoors, but online optimization, dynamics, and large-scale deployment remain difficult.
Kimera [24, 29]	Strong	Moderate	Moderate	Strong with VIO	Moderate	Adds semantic and metric structure, but performance depends on the quality and cost of segmentation and mesh/graph construction.

5.7. Key Trade-Offs and Selection Guidelines

The comparison suggests several recurring trade-offs.

5.7.1. Accuracy Vs Efficiency

Learning-based systems have low ATE error as reported in benchmark settings. DROID-SLAM [15] reported 0.022 m of mean ATE on EuRoC and 0.038 on TUM-RGBD datasets, as shown in Table 2. This accuracy is achieved by the use of a GPU and less transparent failure diagnosis. On the contrary, ORB-SLAM3, SVO and VINS-Mono can be deployed on CPU-based platforms and expose clearer residuals, but under conditions such as severe blur, low texture or dynamic foregrounds.

5.7.2. Map Utility Vs Resource Requirements

Sparse maps are sufficient for localization and often efficient. Dense and neural implicit maps support rich interaction, inspection and visualization, but increase the computational cost and memory. Nice-SLAM [20] improves over a single global Multi-Level Perceptron (MLP) by using hierarchical representation, but is not equivalent to low power embedded SLAM solution. Dense mapping, though, appears more advanced and should not be selected unless the application needs surface-level reconstruction.

5.7.3. Robustness Vs Interpretability

The source of errors for geometric systems is through residuals, feature matches, reprojection error, loop constraints and Jacobians. Learned systems, on the other hand, may absorb complex visual patterns and improve robustness, but it is hard to inspect the internal reasoning. This becomes a serious concern for safety-critical applications, where the systems should not only perform well on benchmarks but, at the same time, should also explain their failure or uncertainty.

5.7.4. Generality Vs Specialization

General-purpose systems like ORB-SLAM3 and VINS-Mono can be adapted to many platforms, while semantic and neural systems show better performance in environments that resemble the one that they were trained during their deployment.

DynaSLAM [30] is useful when the dynamic objects match known semantic classes, but its performance lowers in mild motion or when the dynamic objects are outside their detected vocabulary. DROID-SLAM [15] shows better generalization across reported benchmarks, but it still requires validation during deployment with unusual cameras, weather, texture or motion regimes.

5.8. Application Specific Recommendations

Table 4 summarizes the practical recommendations for SLAM systems according to the application conditions, with reasoning and caution.

Table 7. Application-oriented method selection guide

Sr. No	Application condition	Preferred methods	Reason	Main caution
1	Structured indoor or outdoor navigation with moderate texture	ORB-SLAM3, VINS-Mono	Mature tracking, loop closure, re-localization, and efficient deployment	Can degrade with blur, weak texture, or dynamic foregrounds
2	Low-power embedded or high-speed UAV operation	SVO 2.0, VINS-Mono	Low latency and motion robustness	Sparse maps and limited semantic understanding
3	Accuracy-focused research benchmark	DROID-SLAM, ORB-SLAM3, VINS-Mono	Strong reported trajectory accuracy and widely used baselines	Hardware and evaluation protocol must be stated clearly
4	Highly dynamic indoor environments	DynaSLAM, semantic extensions of ORB-SLAM, and Kimera	Dynamic object masking and semantic structure reduce map corruption	Unknown moving objects and segmentation latency remain failure cases
5	Dense indoor reconstruction	NICE-SLAM, iMAP, RGB-D dense SLAM systems	Dense or continuous maps support inspection and visualization	GPU and RGB-D dependence; limited large-scale outdoor suitability
6	Spatial reasoning and planning	Kimera, 3D dynamic scene graph systems	Provides metric, semantic, and hierarchical scene structure	Higher system complexity and dependence on semantic perception
7	AR/VR with strict latency	ORB-SLAM3, SVO-style systems	Fast tracking and re-localization are more important than dense neural mapping.	Textureless surfaces and lighting changes need additional handling.

5.9. Evolutionary Trajectory and Future Directions

SLAM systems evolved from filtering to key-frame based optimization, from sparse features to direct and semi-direct photometric alignment, from geometric-only maps to semantic and object-level maps, and from explicit maps to differentiable and neural implicit representation. Each stage solved some limitations of the previous stage but introduced some new limitations. Direct methods reduced dependence on sparse descriptors but are sensitive to photometric assumptions. Semantic methods improved the way dynamic scenes were handled, but introduced segmentation errors and latency. Learned methods improved dense correspondence but

are less interpretable and increased the demand for hardware. Neural implicit methods became rich in mapping but were difficult to scale and deploy on low-power platforms. Therefore, the future direction is not the replacement of geometric systems by learning-based SLAM systems, but seamless integration of the two systems. Future SLAM systems should combine the efficiency and interpretability of geometric methods with the robustness of the learned perception. This has to be done by preserving uncertainty estimates and failure detection.

6. Major Challenges and Open Problems in Visual SLAM

The comparative analysis in Section 5 shows that modern visual SLAM has reached high accuracy on several standard benchmarks, especially when the camera motion is moderate, the environment is sufficiently textured, and the sensor configuration matches the assumptions of the algorithm. However, the same analysis also shows that strong benchmark performance does not automatically translate into dependable long-term autonomy. A method may report low ATE on EuRoC, TUM-RGBD, or TartanAir, yet still fail when the scene is crowded, the lighting changes abruptly, or the camera is poorly calibrated, the deployment platform has limited compute, or the map must be maintained for months. Therefore, the main open problems are no longer only about reducing trajectory error under controlled conditions. It is poorly calibrated, about how visual SLAM can be made robust, interpretable, scalable, resource-aware and adaptable to the real world. Then it will be considered as a consistent shift towards the age of robust perception of SLAM [40].

The subsection below provides a few challenges that are connected closely to the tradeoff identified earlier. Feature-based methods interpretable and reliable, but they often fail when reliable features are not available [11, 13]. On the other hand, direct methods are capable of using image intensities but, in turn, have to depend on photometric consistency and proper calibration of cameras [14]. Visual-inertial systems provide better robustness in motion but areaccompanied by initialization, synchronization and calibration constraints [22, 23]. Semantic and object-level systems provide better scene awareness, but they have to depend on segmentation quality and show an increase in computation time [24, 28-30, 60]. Learned and neural implicit methods provide dense tracking and mapping, but they seldom generalize, depend on GPU and may be uncertain [15, 19, 20, 72]. These systems, along with their limitations, help to define current challenges.

6.1. Robustness in Challenging and Dynamic Environments

Robustness remains the most visible barrier to using visual SLAM in unconstrained environments. Many systems still rely, explicitly or implicitly, on assumptions that the observed world is mostly static, sufficiently textured, and repeatedly visible. These assumptions are reasonable for many laboratory sequences but weak for hospitals, warehouses, roads, construction sites, farms, underwater scenes, and crowded public spaces.

Highly dynamic and crowded scenes are a major failure case. Conventional geometric systems may track features on moving people, vehicles, screens, doors, or furniture and then insert those points into a static map. Once this happens, pose estimation and loop closure become unreliable because the map no longer represents a fixed structure. DynaSLAM addresses this problem by masking dynamic objects through semantic segmentation and multi-view geometry [30], while more recent approaches, such as Panoptic-SLAM and VAR-SLAM, continue to improve robustness in dynamic scenes by using panoptic or adaptive motion-aware reasoning [33, 34]. However, dynamic scene SLAM is still not solved. Class-based segmentation may miss unknown moving objects, transparent objects, deformable surfaces, vegetation, water, reflections, or partially occluded humans. A system may also remove too much of the image, leaving insufficient static background for tracking. The open problem is not simply to detect motion, but to decide which parts of the scene are stable enough for mapping, which parts should be tracked as dynamic entities, and which parts should be ignored.

Adverse visual conditions create a second layer of difficulty. Severe illumination change, direct sunlight, night operation, motion blur, rolling shutter distortion, rain, fog, snow, dust, and repetitive textures can all reduce the quality of feature matching or photometric alignment. On reported benchmarks, learned systems such as DROID-SLAM [15] are found to be more robust than classical algorithms. To achieve robustness, the algorithm has to be trained, which in turn adds hardware cost. Direct methods such as DSO [14], make effective use of image gradient, but their performance can degrade. This happens when the basic assumption of the algorithm, such as calibration, brightness constancy, or exposure, is violated. NetVLAD [73], an algorithm based on place recognition, improves loop closure but may cause false negative or false positive results on large structural or seasonal changes.

The challenge before the current systems is to recognize that the visual evidence they receive is becoming unreliable. The front end of the future SLAM systems should be able to estimate the confidence in their measurements along with their conventional strengths. To deliver robustness, the SLAM systems will need awareness of uncertainty, adaptive to outlier rejection reason dynamic objects and should be aware of the failure during re-localization.

6.2. Scalability, Efficiency, and Lifelong Operation

Scalability is a broad umbrella that encompasses the processing of long sequences, as it is one of the parts. If the

robot happens to operate for weeks or years, it must be able to decide more than the ability to process a long sequence. A robot operating for weeks or years must decide which data to store, update, compress, or erase. If the system retains every keyframe, land mask, semantic label or neural parameter throughout its working, then the optimization and memory cost will grow enormously. If too much information is removed, the system loses re-localization ability and global consistency.

ORB-SLAM3 makes an important step through its Atlas representation, which supports multiple maps and improves recovery after tracking loss [13]. Pose-graph optimization, factor graphs, and solvers such as g2o and GTSAM provide the mathematical tools for scalable estimation [7, 44, 45, 47]. However, large-scale lifelong SLAM still faces unresolved map-management questions. The system must handle revisited places, structural renovations, moved furniture, changed lighting, newly added objects, and areas that disappear from view for long periods. A warehouse shelf, for example, may be a reliable landmark in one month and a misleading landmark in the next. Current systems often either keep stale map information or start a new map, but long-term autonomy requires a more selective and evidence-based update mechanism.

Efficiency is coupled with scalability. Section 5 showed that methods such as ORB-SLAM3, SVO, and VINS-Mono are more practical for CPU-based deployment, while DROID-SLAM and neural implicit methods require heavier computation [13, 15, 20, 22, 70]. The problem becomes more serious when semantic segmentation, dense meshing, object-level mapping, or neural rendering is added. Kimera and 3D Dynamic Scene Graphs show how useful metric-semantic representations can be for spatial reasoning [24, 27, 29], but they also illustrate that richer maps require richer computation.

Future SLAM systems need maps that are task-aware and hierarchical in nature. Depending upon the application, the map density should change. If dealing with a navigation system, then it may require a sparse and globally consistent graph, while systems used for inspection may require dense local geometry. Recent work shows passage-aware RGB-D SLAM systems [35] that provide maps that can encode architectural and topological elements that are useful, instead of building maps on dense geometry. Scalable SLAM systems do not keep adding points or neural parameters, it must be able to preserve the most important ones.

6.3. Sensor Fusion and Calibration

Sensor fusion improves robustness by combining complementary information. Cameras provide rich appearance and geometry, IMUs provide high-rate motion constraints, depth cameras provide metric structure, event cameras handle high-speed and dynamic range scenes, and LiDAR can provide reliable range in weak-texture environments. Visual-inertial methods such as OKVIS, VINS-Mono, OpenVINS, and ORB-SLAM3 show the value of tightly coupled camera-IMU estimation [13, 21-23, 65]. Event camera fusion also shows promise for high-speed and high-dynamic-range operation [67].

The difficulty is that sensor fusion is only as reliable as the calibration and timing behind it. A small error in camera-IMU extrinsic, rolling-shutter timing, IMU bias, depth scale, or timestamp synchronization can appear as drift, inconsistent gravity, distorted maps, or failed initialization. Visual-inertial systems can estimate some calibration parameters online, but this does not make calibration a solved problem [13, 22, 65]. Online calibration may become weakly observable during low-motion sequences, may drift when visual tracking is poor, or may fail after mechanical vibration and sensor displacement.

Heterogeneous fusion is an even harder problem. Many current systems add sensor measurements as separate factors in an optimization graph. This is mathematically clean, but it does not always solve the deeper issue of how to fuse different kinds of information at the representation level. Uncertainties and failure modes are different for camera images, event streams, semantic masks, IMU pre-integration and depth maps. If they are treated independently or equally reliable, they can overfit the SLAM systems. The open challenge is to build a fusion system that would independently decide which sensor modality should dominate depending upon the scenario, or if either of the modality in the system is failing.

Future sensor fusion research should therefore focus on self-calibration, temporal synchronization, uncertainty-aware weighting, and lifelong calibration monitoring. A practical robot should be able to detect that a camera has shifted, an IMU has developed bias, a depth sensor is noisy, or a semantic model is unreliable in the current scene. Without this ability, multi-sensor SLAM can appear robust while silently accumulating inconsistent constraints.

6.4. Semantic and Object-Level Understanding

The move from geometric maps to semantic and object-level maps is necessary for robots that must interact with the world rather than only localize within it. A sparse point cloud may be enough for tracking, but it does not tell a robot where the door is, whether an object is movable, which region is traversable, or how rooms are connected. Semantic systems such as Kimera, 3D Dynamic Scene Graphs, CubeSLAM, MID-Fusion, and SO-SLAM show how objects, meshes, labels, and higher-level structure can be integrated into SLAM [24, 26-29, 60].

The main challenge is that semantic information is uncertain and changes over time. Object detectors and segmentation networks can misclassify objects, miss small instances, merge adjacent objects, or change predictions across frames. If those predictions are inserted into the SLAM

map without uncertainty, the system may build a semantically inconsistent map even when the geometry is accurate. Dynamic objects add another complication: a chair, trolley, or hospital bed may be movable but still useful as a short-term landmark; a person should usually not be used as a static landmark; a door may be static in one frame but change state later. Visual SLAM systems should be semantic (temporal, action-aware and probabilistic) in nature.

Another problem is bundle adjustment at the object level. CubeSLAM [60] and SO-SLAM [28] are able to detect objects of simple and symmetric shapes. This limits the object detection in real-life cases where objects with vast diversity and scale are found. Semantic SLAM systems need to develop generalize object detection and representation estimation as well as reasoning without being too expensive.

The next milestone is to combine features such as geometry, topology, semantic and affordance. An example of this is Passage Aware mapping [35] which treats doors and openings as structural elements and not just pixels or points relevant for mapping. 3D Dynamic Scene Graphs [27] uses places, objects and humans as nodes and then places a hierarchical relationship rather than a pure geometric map. The challenge now is to maintain these rich points more efficiently and accurately throughout the course of SLAM operation.

6.5. Limitations of Neural and Implicit Representations

Neural and implicit representations such as iMAP [19] and NICE-SLAM [20] represent scenes as continuous learned functions or hierarchical neural fields instead of sparse landmarks or dense voxels. DROID-SLAM [15] shows the way to improve the accuracy of the trajectory in benchmark settings through learned dense correspondence and recurrent optimization. Uncertainty learning incorporated by UncLe-SLAM [72] improves the performance of dense neural SLAM on indoor benchmarks, but they are accompanied by some practical limitations.

Computational cost is the first limitation of neural implicit networks. They require GPU acceleration, repeated optimization and careful memory management. It is suited for research labs or high-end robots, but difficult for systems that work on low-power mobile robots, drones and AR/VR headsets. The real-time behavior of neural implicit methods depends on resolution, GPU class, update frequency, and scene size. During its development, the question is whether it will run while performing tasks such as perception, planning, control, communication and safety management.

Scalability is the second limitation of neural implicit methods. Earlier, these systems performed well for a room-scale environment. By using hierarchical representation, NICE-SLAM [20] provides an improvement in scalability. However, if the environment is outdoor, city scale, dynamic or lifelong, then it becomes difficult to perform. Dynamic SLAM systems such as NID-SLAM [36] and DDN-SLAM [74] can handle dynamic objects but face challenges in maintaining their learned maps over a long period without high computation.

Interpretability and verification are the third limitation of neural implicit systems. They provide excellent output, but their failure is difficult to diagnose. This becomes a major concern for systems such as safety-critical robots, autonomous driving, medical navigation and industrial inspection. Generally, the systems must be able to communicate uncertainty, provide failure signals; otherwise, the user may depend upon plausible maps or wrong trajectories.

The open problem for neural SLAM is to make it accountable. To make them accountable, it has to combine learned components with geometry, residuals that are interpretable, uncertainties that are calibrated and a fallback mechanism. Neural SLAM needs to be embedded in systems that can bound or detect their failure modes to become useful.

6.6. Generalization and the Simulation to Real Gap

Learning-based visual SLAM systems depend on the data on which their models are trained. This data may be synthetic indoor data, clean RGB-D scans, or curated benchmark sequences. The SLAM systems trained on such data may not perform well in real-life scenarios such as rural roads, underwater scenes, dusty warehouses or agricultural fields. This problem surfaces when the system performs feature matching, depth estimation, optical flow, segmentation of dynamic objects and neural mapping.

Generalization of the domain is the core challenge. TartanAir [75] was designed for a difficult and adverse environment. Though DROID-SLAM [15] performed well on benchmark datasets, it may not perform similarly upon deployment. Systems should be evaluated on failure rate, recovery behavior, uncertain calibration and performance on controlled domain changes along with mean ATE.

The gap between simulation to real is closely related. Simulation provides ground truth with accuracy, generate large scale data and controlled dynamics. It is very important during training and evaluation because the ground truth in real time is expensive or sometimes impossible. However, the images that we obtain from simulation differ in texture, lightning, motion blur, rolling shutter, sensor noise, reflectance of the material and scene complexity. Even though SLAM systems learn during training and perform well in simulation, they fail in real-life scenarios.

Future systems will require certain protocols for evaluation that can combine simulation and real-world datasets. EuRoC [76], TUM-RGBD [77], TartanAir [75], and multi-view stereo benchmarks [78] datasets will remain important because different SLAM systems can be compared with their use. However, future datasets should consider

sensor degradation, long-term revisits, dynamic scenes, adverse weather, semantic change, texture with low spaces and calibration perturbation.

7. Leading Open-Source Libraries & Frameworks in State-of-the-Art Visual SLAM

Advancement in visual SLAM algorithms is possible due to the development of robust open-source libraries. These libraries provide a benchmark for visual SLAM algorithms and enabled it use by academicians, researchers and industry.

The framework of these libraries encapsulates the main trends such as optimization, multisensor fusion, and semantic understanding.

7.1. Geometric Optimization Workhorses

These libraries provide a backbone to recent visual SLAM systems. At the core of SLAM systems, they provide solvers for nonlinear least square problems.

- g2o (General Graph Optimization) [44]: It is a C++ framework that is used to solve nonlinear least square optimization problems. In SLAM systems, it refines the poses and map landmarks by optimizing all constraints together in a graph-like structure. It is highly flexible and has been the default optimizer for countless SLAM systems, including the early versions of ORB-SLAM.
  Key Characteristic: Provides a collection of solvers, namely Gauss-Newton, Levenberg-Marquardt and a variety of nonlinear solvers.
- Ceres Solver [79]: A powerful and widely used open-source C++ library developed by Google, which is used for solving large-scale optimization problems. Its robust implementation of non-linear least-squares optimization makes it a popular choice for Bundle Adjustment.
  Key Characteristic: Features built-in automatic differentiation, eliminating the need to compute Jacobians manually.
- GTSAM (Georgia Tech Smoothing and Mapping) [80]: A BSD-licensed C++ library that uses factor graphs, which implements smoothing and mapping with Gaussian distributions. It emphasizes a probabilistic approach and is particularly strong in incremental inference and iSAM2 (Incremental Smoothing and Mapping).
  Key Characteristic: Uses a full factor graph representation, making it intuitive to model complex multi-sensor problems.

7.2. Complete End-to-End SLAM Systems

These are comprehensive, ready-to-use pipelines that leverage recent algorithms in specific domains.

- ORB-SLAM Series (1, 2, 3): Arguably the most influential and complete open-source SLAM systems built on feature-based methods. ORB-SLAM3 [13] supports monocular, stereo, as well as RGB-D cameras, visual-inertial odometry, and multi-map lifelong operation.
  Key Characteristic: A full-featured system which enables tracking, local and global mapping, and loop closing, all built around the ORB feature and a covisibility graph.
- OpenVINS [65]: It is an open-source platform for visual-inertial estimation research that implements a modular filtering-based pipeline, providing a comparison point to the dominant optimization-based approaches.
  Key Characteristic: Offers a highly configurable EKF-based VIO pipeline with a suite of simulation and evaluation tools.
- Kimera [29]: A C++ library for real-time semantic SLAM. It contains a modular set of components for visual-inertial SLAM (Kimera-VIO), dense 3D mesh reconstruction (Kimera-Mesher), and semantic label fusion (Kimera-Semantics).
  Key Characteristic: One of the most cited systems to tightly integrate real-time geometric SLAM with semantic segmentation to build a semantic mesh.
- Basalt (BAse SAmpler for Lightweight Tracking) [81]: A high-performance, optimization-based visual-inertial odometry system. It is renowned for its accuracy and efficiency, using a direct frontend with a spline-based trajectory representation.
  Key Characteristic: Uses a direct photometric frontend combined with a spline-based continuous-time backend for VIO.

7.3. Learning-Enhanced and Differentiable Frameworks

This category represents progressive frameworks where deep learning is integrated into the SLAM pipeline.

- DROID-SLAM [15]: In DROID-SLAM, the framework is built on a neural iterative optimizer that jointly estimates camera poses, dense depth maps, and dense optical flow. It replaces the classical geometric bundle adjustment with a learned recurrent Gauss-Newton optimizer that iteratively updates dense correspondences and poses.
  Key Characteristic: It is fully differentiable and demonstrates high performance in low-texture and motion-blurred environments.
- DPVO (Differentiable Particle Filter Visual Odometry) [82]: The framework is a fully differentiable particle filter, which is designed for real-time high accuracy.
  Key Characteristic: It is a multimodal, learnable, and robust framework.

7.4. Supporting Libraries and Tools

These are the critical components that are used by nearly all modern SLAM frameworks.

- OpenCV (Open Source Computer Vision Library) [83]: OpenCV is the primarily used software library for the visual SLAM pipeline. It performs feature extraction and matching, camera calibration, triangulation, 3D

reconstruction, optical flow reconstruction, image preprocessing and enhancing, and visualization.

- DBoW2 (Bag of Binary Words for Place Recognition) [84]: DBoW2 is a C++ library used for the creation of visual vocabularies and retrieving images using bag-of-words. In SLAM systems, it is used for keyframe retrieval, re-localization and loop closure detection.
- Sophus [85]: Sophus is a C++ template library that implements Lie Groups and Lie Algebras, finds its use in robotics, computer vision and SLAM. It correctly handles 3D rotations, translations, and pose updates using Lie groups.

7.5. Summary and Impact

These open source libraries have made visual SLAM accessible to everyone. They have provided the following:

1. Benchmarks: Systems like ORB-SLAM3 provide a reference performance comparison of SLAM systems.
2. Modularity: Libraries like GTSAM and Kimera allow researchers to replace one component of SLAM with another.
3. Educational Value: Well-documented codebases such as OpenVINS are extremely useful for novice researchers.
4. Accelerated Development: Open source libraries allow practitioners to build their own applications without rewriting complex mathematical and algorithmic functions.

The continued development in these frameworks, especially the progress in learning-based approaches like DROID-SLAM, will be pivotal for meeting the upcoming challenges in robust long-term autonomy.

8. Future Outlook

Visual SLAM has evolved from geometric consistency achieved in controlled settings to developing a spatial perception system. These systems are reliable and useful for autonomous agents, helping them to navigate as well as interact with the real world. The SLAM system has matured from handcrafted geometry to learning-based systems, whereas the maps have evolved from sparse and short-term to dense and long-term.

The current systems consist of a diverse set of algorithms tailored for various constraints rather than just a single algorithm. ORB-SLAM series reflects the way in which the geometric methods have changed, whereas DROID-SLAM provides robustness in learned optimization. iMAP and NICE-SLAM explain how the neural scene representation helps to reshape the SLAM pipeline.

8.1. Future Research Challenges in Visual SLAM

The preceding section marks the progress of visual SLAM from sparse geometric mapping to learned correspondence, visual-inertial fusion, semantic mapping and neural implicit representation. However, the main challenge is to make visual SLAM more dependable when there is a change in environment, hardware budget, sensor quality and the task assigned. Future research should be more focused on systems that are more scalable, robust, interpretable, computationally efficient and able to produce useful maps for a longer period of time.

8.1.1 Robustness Beyond Controlled Benchmarks

A future challenge for the visual SLAM system is to become more dependable beyond controlled benchmark settings. It is seen from the system such as ORB-SLAM3 [13], DSO [14], VINS-Mono [22], and DROID-SLAM [15]. VINS-Mono performs well in controlled benchmark settings but still shows signs of failure mode.

Feature-based methods may show limited performance in low texture, great light changes, blur or repeated structures, while direct methods may show degraded performance when their photometric assumption is broken. Learned methods, though robust, require careful validation for unfamiliar cameras and the conditions for their performance.

Future work should be aware of rather than just average case performance. The systems should be able to track failure rates, fetch recovery time after failure, calibrate uncertainty, show success in re-localization and show performance under controlled degradation such as blur, low light, fog, rain glare and low texture. This would make it easier to distinguish methods that are merely accurate on successful runs from methods that remain dependable under stress [75-78].

8.1.2 Dynamic Scene SLAM without Over-Relying on Known Object Classes

Dynamic environments remain a major open challenge, although DynaSLAM [30] showed that masking moving objects can greatly improve performance in highly dynamic TUM-RGBD sequences; as discussed in Section 5, this benefit depends on the scene, since semantic masking is useful when moving foregrounds dominate but may remove useful information in mildly dynamic scenes.

Recent methods such as VAR-SLAM [34] and Panoptic-SLAM [33] extend this direction through adaptive and panoptic motion-aware reasoning. However, future systems must handle motion beyond fixed object categories, including moving furniture, doors, screens, curtains, water, vegetation, shadows, reflections, and deformable objects.

Future SLAM systems should decide whether a region is stable enough for mapping based on geometry, motion consistency, temporal persistence, semantic cues, and uncertainty together. A useful research direction is to represent the world as a mixture of static structure, temporarily stable objects, and actively dynamic entities, rather than treating every pixel as either static or dynamic.

8.1.3. Confidence-Aware Front-Ends and Failure Detection

Most SLAM pipelines still treat front-end outputs such as

feature matches, optical flow, semantic masks, and depth estimates as similarly reliable, even though a match near a moving boundary, a mask under poor lighting, or a depth estimate on a reflective surface is far less trustworthy than a stable landmark observed across many frames.

Future frontends should produce both measurements and confidence. This applies to handcrafted features, learned features, deep optical flow, semantic segmentation, object detection, and neural depth. These confidence estimates should then influence outlier rejection, factor weighting, keyframe selection, loop closure, and re-localization. Work on uncertainty learning for dense neural SLAM, such as UncLe-SLAM, points toward this direction [84], but uncertainty still needs to be connected more tightly with classical geometric residuals, map maintenance, and failure recovery.

8.1.4. Lifelong Map Management

Long-term operation requires more than simply building a large map, because a robot working for months must decide what information to keep, update, compress, or discard, and although ORB-SLAM3's [13] Atlas supports multi-map operation and recovery after tracking loss while graph-based optimization provides scalable tools [7, 44, 47, 80], reliable long-term map management remains difficult.

Future research should address map ageing, since some map elements become unreliable as objects move, buildings change, lighting varies, or seasonal appearance shifts, while other elements remain stable and should be preserved. A future SLAM system should store temporal validity, observation history, uncertainty, and semantic meaning for each map element, so it can judge not only that a landmark exists but also when it was reliable and whether it now needs verification, especially in gradually changing places such as warehouses, hospitals, homes, agricultural fields, and public spaces.

8.1.5. Scalable Dense and Neural Representations

Neural implicit methods such as iMAP [19] and NICE-SLAM [20] have broadened SLAM maps from sparse landmarks to continuous dense scene representations, while DROID-SLAM [15] and DPVO [82] show the value of learned dense correspondence and optimization, but these dense and neural maps still demand high memory, computation, and energy, making them difficult to scale to large outdoor and long term settings.

Future work should focus on hybrid map representations, where a practical SLAM system uses a sparse global graph for long-range consistency, local dense geometry for manipulation or inspection, semantic objects for planning, and neural fields only where high fidelity reconstruction is truly needed. Hierarchical methods, multiresolution grids, hash encodings, tensor factorizations, and structural abstractions can reduce dense mapping cost [20, 35, 68, 69], but the key challenge is choosing the right representation for each part of the environment while updating the map without losing global consistency.

8.1.6. Semantic Maps that are Temporally Consistent

Semantic SLAM should go beyond attaching labels to geometry and instead maintain a consistent scene understanding over time, since although Kimera [24, 29], 3D Dynamic Scene Graphs [27], CubeSLAM [60], MID-Fusion [26], and SO-SLAM [28] show strong progress in metric-semantic and object-level mapping, object detectors and segmentation networks can still produce inconsistent predictions across frames under occlusion, viewpoint changes, poor lighting, or domain shift.

Future semantic SLAM should treat labels as uncertain and time-dependent, updating object identity, class, shape, pose, and motion state as new evidence becomes available. It should reason when the states of the object change, such as the opening of doors, moving chairs, and people passing. This would require tighter integration of geometric estimation, semantic inference, object tracking and reasoning of scene graphs.

8.1.7. Object-Level Bundle Adjustment

Point landmarks are better handled by classical bundle adjustment [43], because its reprojection error [46], can be clearly defined. However, in object-level SLAM, objects are represented in many ways, such as boxes, planes, quadrics, meshes, keypoints, and symmetry constraints, which limits the use of the full potential of bundle adjustment. Although CubeSLAM [60] and SO-SLAM [28] show that even in the presence of object constraints, mapping can be improved by using simplified object models. A key future challenge is to develop object-level optimization that will be general and computationally efficient. The model should work while handling partial views, occlusions, object symmetry, scale ambiguity, change in object pose and uncertain semantic labels.

8.1.8. Sensor Fusion with Self-Calibration and Reliability Monitoring

Visual-inertial SLAM works by combining IMU and cameras, which add the strength of each modality. IMU supports tracking even during fast motion, blur and visual failures [21-23], while event cameras improves robustness in high-speed and high-dynamic scenes [67]. However, sensor fusion can still fail due to improper calibration, synchronization and changes in sensor reliability.

Future sensor fusion should be able to detect camera-IMU time offsets, drift in IMU bias, sensor degradation, depth scale errors and rolling shutter effects. It should give more importance to visual sensors when visual textures are prominent and to IMU when the motion is fast or in the presence of low-light scenes. The key challenge in sensor

fusion is for switching of sensors should be principled rather than heuristics.

8.1.9. Interpretable and Verifiable Learning-based SLAM

Learning-based SLAM methods such as DROID-SLAM [15] and UncLe-SLAM [72] have achieved high accuracy. These systems should indicate uncertainty, detect untrained input data and know when the mechanism fails.

Future research should provide explicit geometric checks with learned components. This would audit, regularize and debug learned perception by using geometry.

8.1.10. Generalization across Domains and Sensors

Domain shift from synthetic or curated benchmarks to generalized scenes such as rural or urban roads, hospitals, factories, and agricultural fields is still a critical challenge. tartanAir [82], a harder synthetic environment, still cannot capture real sensor noise, material reflectance, weather, blur or geographic scene difference.

Future evaluations should include models that are assessed on cameras, environment and motion patterns that differ from training. Reports should clearly mention training data, testing data, hardware used, failure cases and overlap with the training data. Hidden data dependence in learned SLAM, semantic SLAM and neural implicit mapping makes these methods seem more general than they really are.

8.1.11. Task-Aware SLAM instead of One-Map-Fits-All Mapping

Different applications need different kinds of maps. A drone may require fast but sparse localization, a warehouse robot may need robust long-term navigation, and an inspection robot may need dense geometry. Future SLAM systems should become task-aware rather than building a rich map everywhere. A task-aware SLAM system should allocate computation according to what the robot needs, maintaining sparse landmarks for global localization, dense local surfaces near obstacles, object-level maps for manipulation, and semantic scene graphs for planning. This direction is supported by metric-semantic and structural mapping approaches such as Kimera, 3D Dynamic Scene Graphs, and passage-aware structural mapping [24, 27, 35], but the key challenge is designing policies that decide what to model, at what resolution, and for how long.

8.1.12. Better Benchmarking and Reporting Standards

The field needs stronger evaluation protocols because, although benchmarks such as EuRoC [76], TUM-RGBD [77], TartanAir [75], and multi-view [78] stereo datasets provide useful foundations, a single mean ATE cannot capture robustness, scalability, runtime, memory use, re-localization, map quality, semantic consistency, or failure recovery, so future evaluations should also include dynamic scenes, adverse weather, long-term revisits, calibration perturbations, sensor failures, low-texture spaces, semantic change, and out-of-distribution settings.

Future papers should report accuracy along with failure rate, runtime, hardware, memory use, energy demand, map size, re-localization success, uncertainty calibration, and clear examples of failure cases. This would make comparisons more transparent and practically useful. This would help researchers to judge whether a method is suitable for embedded robots, AR/VR, autonomous vehicles or long-term service robots.

8.2. Emerging Application Domains

Visual SLAM systems will give rise to new application domains as they mature.

- Complex dynamic environments such as warehouses, hospitals and public places for autonomous mobile robots.
- In extended reality (XR), the SLAM system has to perform robustly and possess low latency for spatial understanding.
- Infrastructure inspection and monitoring for autonomous drones and ground robots.
- use in personalized robotics for domestic and healthcare applications.
- use for environmental monitoring and conservation efforts in natural habitats.

9. Conclusion

Visual SLAM developed from a mainly geometry-driven estimation problem into a broader spatial perception framework that combines geometry, learning, semantics, dense mapping, and sensor fusion. This review manifests that classical methods still remain highly relevant because they provide clear mathematical structure, computational efficiency, and interpretability. At the same time, learning-based and neural approaches have expanded the capability of SLAM systems, particularly in difficult conditions involving weak texture, motion blur, dense correspondence, semantic understanding, and richer scene representation. It is observed from the reviewed literature that there is no single visual SLAM approach that is suitable for every application. Visual SLAM with feature-based methods, direct methods, visual-inertial, object-level, semantics, differential and neural implicit methods have their own strengths and limitations. Therefore selection of a particular type of visual SLAM architecture will depend upon the operating environment, computational resources, sensor configuration and map requirement. For real-world deployment, there should be a good balance between accuracy, robustness, ability to re-localize, uncertainty awareness, runtime cost and maintenance of long-term maps.

In the near future, visual SLAM will likely shift to a hybrid system instead of a complete replacement of geometry with learning. Geometry will still provide consistent pose estimation and optimization, while learning will improve

adaptability, semantic reasoning and dense scene understanding. The progress of SLAM systems will depend upon systems that can recognize unreliable measurements, handle dynamic and changing environments, long-term map management and reporting of uncertainty in a meaningful way. In the current scenario, visual SLAM is progressing towards dependable, task-aware and spatial intelligent systems. Continued development will be important for autonomous vehicles, mobile robots, XR systems, drones, healthcare robotics and robots operating in complex environments. The main challenge is to build SLAM systems that are accurate, reliable, interpretable, and that can be deployed in actual real scenarios.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

References

[1] Randall C. Smith, and Peter Cheeseman, "On the Representation and Estimation of Spatial Uncertainty," *The International Journal of Robotics Research*, vol. 5, no. 4, p. 56-68, 1986. [CrossRef] [Google Scholar] [Publisher Link]

[2] Randall Smith, Matthew Self, and Peter Cheeseman, *Estimating Uncertain Spatial Relationships in Robotics*, Autonomous Robot Vehicles, Springer, New York, NY, pp. 167-193, 1990. [CrossRef] [Google Scholar] [Publisher Link]

[3] H. Durrant-Whyte, and T. Bailey, "Simultaneous Localization and Mapping: Part I," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99-110, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[4] T. Bailey, and H. Durrant-Whyte, "Simultaneous Localization and Mapping (SLAM): Part II," *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 108-117, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[5] Sebastian Thrun, Wolfram Burgard, and Dieter Fox, *Probabilistic Robotics*, Intelligent Robotics and Autonomous Agents Series, The MIT Press, pp. 1-668, 2005. [Google Scholar] [Publisher Link]

[6] Michael Montemerlo et al., "FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges," *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, Acapulco Mexico, pp. 1151-1156, 2003. [Google Scholar] [Publisher Link]

[7] F.R. Kschischang, B.J. Frey, and H.A. Loeliger, "Factor Graphs and the Sum-Product Algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498-519, 2001. [CrossRef] [Google Scholar] [Publisher Link]

[8] Sebastian Thrun, and Michael Montemerlo, "The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures," *The International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 403-429, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[9] Andrew J. Davison et al., "MonoSLAM: Real-Time Single Camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052-1067, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[10] Georg Klein, and David Murray, "Parallel Tracking and Mapping for Small AR Workspaces," *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, Nara, Japan, pp. 225-234, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[11] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147-1163, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[12] Raúl Mur-Artal, and Juan D. Tardós, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255-1262, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[13] Carlos Campos et al., "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874-1890, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[14] Jakob Engel, Vladlen Koltun, and Daniel Cremers, "Direct Sparse Odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 611-625, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[15] Zachary Teed, and Jia Deng, "DROID-SLAM: Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras," *Advances in Neural Information Processing Systems*, pp. 16558-16569, 2021. [Google Scholar] [Publisher Link]

[16] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich, "SuperPoint: Self-Supervised Interest Point Detection and Description," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, Salt Lake City, UT, USA, pp. 337-349, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[17] Paul-Edouard Sarlin et al., "SuperGlue: Learning Feature Matching with Graph Neural Networks," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, pp. 4937-4946, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[18] Krishna Murthy Jatavallabhula et al., "gradSLAM: Automagically Differentiable SLAM," *arXiv Preprint*, pp. 1-13, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[19] Edgar Sucar et al., "iMAP: Implicit Mapping and Positioning in Real-Time," *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, Montreal, QC, Canada, pp. 6209-6218, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[20] Zihan Zhu et al., "NICE-SLAM: Neural Implicit Scalable Encoding for SLAM," *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, New Orleans, LA, USA, pp. 12776-12786, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[21] Stefan Leutenegger et al., "Keyframe-based Visual–Inertial Odometry using Nonlinear Optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314-334, 2015. [CrossRef] [Google Scholar] [Publisher Link]
[22] Tong Qin, and Peiliang Li, "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004-1020, 2018. [CrossRef] [Google Scholar] [Publisher Link]
[23] Christian Forster et al., "On-Manifold Preintegration for Real-Time Visual--Inertial Odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1-21, 2017. [CrossRef] [Google Scholar] [Publisher Link]
[24] Antoni Rosinol et al., "Kimera: From SLAM to Spatial Perception with 3D Dynamic Scene Graphs," *The International Journal of Robotics Research*, vol. 40, no. 12-14, pp. 1510-1546, 2021. [CrossRef] [Google Scholar] [Publisher Link]
[25] Ben Mildenhall et al., "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99-106, 2021. [CrossRef] [Google Scholar] [Publisher Link]
[26] Binbin Xu et al., "MID-Fusion: Octree-based Object-Level Multi-Instance Dynamic SLAM," *2019 International Conference on Robotics and Automation (ICRA)*, Montreal, QC, Canada, pp. 5231-5237, 2019. [CrossRef] [Google Scholar] [Publisher Link]
[27] Antoni Rosinol et al., "3D Dynamic Scene Graphs: Actionable Spatial Perception with Places, Objects, and Humans," *arxiv Preprint*, pp. 1-11, 2020. [CrossRef] [Google Scholar] [Publisher Link]
[28] Ziwei Liao et al., "SO-SLAM: Semantic Object SLAM with Scale Proportional and Symmetrical Texture Constraints," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4008-4015, 2022. [CrossRef] [Google Scholar] [Publisher Link]
[29] Antoni Rosinol et al., "Kimera: An Open-Source Library for Real-Time Metric-Semantic Localization and Mapping," *2020 IEEE International Conference on Robotics and Automation (ICRA)*, Paris, France, pp. 1689-1696, 2020. [CrossRef] [Google Scholar] [Publisher Link]
[30] Berta Bescos et al., "DynaSLAM: Tracking, Mapping, and Inpainting in Dynamic Scenes," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4076-4083, 2018. [CrossRef] [Google Scholar] [Publisher Link]
[31] Olaya Álvarez-Tuñón, Yury Brodskiy, and Erdal Kayacan, "Monocular Visual Simultaneous Localization and Mapping: (R)Evolution from Geometry to Deep Learning-Based Pipelines," *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 5, pp. 1990-2010, 2024. [CrossRef] [Google Scholar] [Publisher Link]
[32] Thanh Nguyen Canh et al., "Semantic Visual Simultaneous Localization and Mapping: A Survey on State of the Art, Challenges, and Future Directions," *Robotics and Autonomous Systems*, vol. 203, pp. 1-37, 2026. [CrossRef] [Google Scholar] [Publisher Link]
[33] Gabriel Fischer Abati et al., "Panoptic-SLAM: Visual SLAM in Dynamic Environments using Panoptic Segmentation," *2024 21st International Conference on Ubiquitous Robots (UR)*, New York, NY, USA, pp. 1-8, 2024. [CrossRef] [Google Scholar] [Publisher Link]
[34] João Carlos Virgolino Soares, Gabriel Fischer Abati, and Claudio Semini, "VAR-SLAM: Visual Adaptive and Robust SLAM for Dynamic Environments," *arxiv preprint*, pp. 1-8, 2025. [CrossRef] [Google Scholar] [Publisher Link]
[35] Ali Tourani et al., "Passage-Aware Structural Mapping for RGB-D Visual SLAM," *arxiv preprint*, pp. 1-5, 2026. [CrossRef] [Google Scholar] [Publisher Link]
[36] Ziheng Xu et al., "NID-SLAM: Neural Implicit Representation-based RGB-D SLAM In Dynamic Environments," *2024 IEEE International Conference on Multimedia and Expo (ICME)*, Niagara Falls, ON, Canada, pp. 1-6, 2024. [CrossRef] [Google Scholar] [Publisher Link]
[37] Lahav Lipson, Zachary Teed, and Jia Deng, "Deep Patch Visual SLAM," *18th European Conference Computer Vision – ECCV, 2024*, Milan, Italy, pp. 424-440, 2024. [CrossRef] [Google Scholar] [Publisher Link]
[38] Ali Rida Sahili et al., "A Survey of Visual SLAM Methods," *IEEE Access*, vol. 11, pp. 139643-139677, 2023. [CrossRef] [Google Scholar] [Publisher Link]
[39] Weifeng Chen et al., "An Overview on Visual SLAM: From Tradition to Semantic," *Remote Sensing*, vol. 14, no. 13, pp. 1-47, 2022. [CrossRef] [Google Scholar] [Publisher Link]
[40] Cesar Cadena et al., "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309-1332, 2016. [CrossRef] [Google Scholar] [Publisher Link]
[41] Konstantinos A. Tsintotas, Loukas Bampis, and Antonios Gasteratos, "The Revisiting Problem in Simultaneous Localization and Mapping: A Survey on Visual Loop Closure Detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 19929-19953, 2022. [CrossRef] [Google Scholar] [Publisher Link]
[42] Stanley F. Schmidt, "Application of State-Space Methods to Navigation Problems," *Advances in Control Systems*, vol. 3, pp. 293-340, 1966. [CrossRef] [Google Scholar] [Publisher Link]
[43] Bill Triggs et al., "Bundle Adjustment — A Modern Synthesis," *Vision Algorithms: Theory and Practice*, Greece, pp. 298-372, 2000. [CrossRef] [Google Scholar] [Publisher Link]
[44] Rainer Kümmerle et al., "G2o: A General Framework for Graph Optimization," *2011 IEEE International Conference on Robotics and Automation*, Shanghai, China, pp. 3607-3613, 2011. [CrossRef] [Google Scholar] [Publisher Link]
[45] Frank Dellaert, "*Factor Graphs and GTSAM: A Hands-on Introduction*," Georgia Institute of Technology, Technical Report, pp. 1-27, 2012. [Google Scholar] [Publisher Link]

[46] Richard Hartley, and Andrew Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed., Cambridge University Press, 2003. [CrossRef] [Google Scholar] [Publisher Link]
[47] Frank Dellaert, and Michael Kaess, "Square Root SAM: Simultaneous Localization and Mapping via Square Root Information Smoothing," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181-1203, 2006. [CrossRef] [Google Scholar] [Publisher Link]
[48] Ethan Rublee et al., "ORB: An Efficient Alternative to SIFT or SURF," *2011 International Conference on Computer Vision*, Barcelona, Spain, pp. 2564-2571, 2011. [CrossRef] [Google Scholar] [Publisher Link]
[49] D. Nister, "An Efficient Solution to the Five-Point Relative Pose Problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 756-770, 2004. [CrossRef] [Google Scholar] [Publisher Link]
[50] Martin A. Fischler, and Robert C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381-395, 1981. [CrossRef] [Google Scholar] [Publisher Link]
[51] Jiexiong Tang, John Folkesson, and Patric Jensfelt, "Geometric Correspondence Network for Camera Motion Estimation," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1010-1017, 2018. [CrossRef] [Google Scholar] [Publisher Link]
[52] Jiexiong Tang et al., "GCNv2: Efficient Correspondence Prediction for Real-Time SLAM," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3505-3512, 2019. [CrossRef] [Google Scholar] [Publisher Link]
[53] Trong Phuc Truong, Vincent Nozick, and Hideo Saito, "Semi-independent Stereo Visual Odometry for Different Field of View Cameras," *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pp. 1-12, 2018. [Google Scholar] [Publisher Link]
[54] David Schubert et al., "Direct Sparse Odometry with Rolling Shutter," *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 682-697, 2018. [Google Scholar] [Publisher Link]
[55] Rui Wang, Martin Schwörer, and Daniel Cremers, "Stereo DSO: Large-Scale Direct Sparse Visual Odometry with Stereo Cameras," *2017 IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, pp. 3923-3931, 2017. [CrossRef] [Google Scholar] [Publisher Link]
[56] Lukas Von Stumberg, Vladyslav Usenko, and Daniel Cremers, "Direct Sparse Visual-Inertial Odometry Using Dynamic Marginalization," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, QLD, Australia, pp. 2510-2517, 2018. [CrossRef] [Google Scholar] [Publisher Link]
[57] S. Gratton, A. S. Lawless, and N. K. Nichols, "Approximate Gauss–Newton Methods for Nonlinear Least Squares Problems," *SIAM Journal on Optimization*, vol. 18, no. 1, pp. 1-25, 2007. [CrossRef] [Google Scholar] [Publisher Link]
[58] Kaiming He et al., "Mask R-CNN," *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 2961-2969, 2017. [Google Scholar] [Publisher Link]
[59] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning*, MIT Press, 2017. [Google Scholar] [Publisher Link]
[60] Shichao Yang, and Sebastian Scherer, "CubeSLAM: Monocular 3-D Object SLAM," *IEEE Transactions on Robotics*, vol. 35, no. 4, pp. 925-938, 2019. [CrossRef] [Google Scholar] [Publisher Link]
[61] Hauke Strasdat, J.M.M. Montiel, and Andrew J. Davison, "Visual SLAM: Why Filter?," *Image and Vision Computing*, vol. 30, no. 2, pp. 65-77, 2012. [CrossRef] [Google Scholar] [Publisher Link]
[62] Niko Sünderhauf, and Peter Protzel, "Switchable Constraints for Robust Pose Graph SLAM," *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura-Algarve, Portugal, pp. 1879-1884, 2012. [CrossRef] [Google Scholar] [Publisher Link]
[63] Gabe Sibley, Larry Matthies, and Gaurav Sukhatme, *A Sliding Window Filter for Incremental SLAM*, Unifying Perspectives in Computational and Robot Vision, Springer, Boston, MA, pp. 103-112, 2008. [CrossRef] [Google Scholar] [Publisher Link]
[64] Tixiao Shan et al., "LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping," *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, NV, USA, pp. 5135-5142, 2020. [CrossRef] [Google Scholar] [Publisher Link]
[65] Patrick Geneva et al., "OpenVINS: A Research Platform for Visual-Inertial Estimation," *2020 IEEE International Conference on Robotics and Automation (ICRA)*, Paris, France, pp. 4666-4672, 2020. [CrossRef] [Google Scholar] [Publisher Link]
[66] Alina Kloss, Georg Martius, and Jeannette Bohg, "How to Train your Differentiable Filter," *Autonomous Robots*, vol. 45, no. 4, pp. 561-578, 2021. [CrossRef] [Google Scholar] [Publisher Link]
[67] Antoni Rosinol Vidal et al., "Ultimate SLAM? Combining Events, Images, and IMU for Robust Visual SLAM in HDR and High-Speed Scenarios," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 994-1001, 2018. [CrossRef] [Google Scholar] [Publisher Link]
[68] Thomas Müller et al., "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding," *ACM Transactions on Graphics*, (TOG), vol. 41, no. 4, pp. 1-15, 2022. [CrossRef] [Google Scholar] [Publisher Link]
[69] Anpei Chen et al., "TensoRF: Tensorial Radiance Fields," *17th European Conference Computer Vision – ECCV 2022*, Tel Aviv, Israel, pp. 333-350, 2022. [CrossRef] [Google Scholar] [Publisher Link]
[70] Christian Forster et al., "SVO: Semidirect Visual Odometry for Monocular and Multicamera Systems," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 249-265, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[71] Yuhao Zhang, Mihai Bujanca, and Mikel Luján, "NGD-SLAM: Towards Real-Time Dynamic SLAM without GPU," *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hangzhou, China, pp. 3467-3473, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[72] Erik Sandström et al., "UncLe-SLAM: Uncertainty Learning for Dense Neural SLAM," *2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, Paris, France, pp. 4539-4550, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[73] Relja Arandjelović et al., "NetVLAD: CNN Architecture for Weakly Supervised Place Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, pp. 1437-1451, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[74] Mingrui Li et al., "DDN-SLAM: Real Time Dense Dynamic Neural Implicit SLAM," *IEEE Robotics and Automation Letters*, vol. 10, no. 5, pp. 4300-4307, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[75] Wenshan Wang et al., "TartanAir: A Dataset to Push the Limits of Visual SLAM," *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, NV, USA, pp. 4909-4916, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[76] Michael Burri et al., "The EuRoC Micro Aerial Vehicle Datasets," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157-1163, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[77] Jürgen Sturm et al., "A Benchmark for the Evaluation of RGB-D SLAM Systems," *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura-Algarve, Portugal, pp. 573-280, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[78] Thomas Schöps et al., "A Multi-View Stereo Benchmark with High-Resolution Images and Multi-Camera Videos," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, pp. 2538-2547, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[79] Sameer Agarwal, and Keir Mierle, "*Ceres Solver: Tutorial & Reference*," Google Inc, 2012. [Google Scholar] [Publisher Link]

[80] Frank Dellaert, and GTSAM Contributors, *Borglab/Gtsam*, Georgia Tech Borg Lab, 2022. [Google Scholar] [Publisher Link]

[81] Vladyslav Usenko et al., "Visual-Inertial Mapping with Non-Linear Factor Recovery," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 422-429, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[82] Zachary Teed, Lahav Lipson, and Jia Deng, "Deep Patch Visual Odometry," *arXiv preprint*, pp. 1-14, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[83] Gary Bradski, "*The Opencv Library*," Dobbs Journal of Software Tools, pp. 1-7, 2000. [Google Scholar] [Publisher Link]

[84] Dorian Galvez-López, and Juan D. Tardos, "Bags of Binary Words for Fast Place Recognition in Image Sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188-1197, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[85] Strasdat/Sophus: C++ implementation of Lie Groups using Eigen, GitHub, 2012. [Online]. Available: https://github.com/strasdat/sophus

[86] Zachary Teed, Lahav Lipson, and Jia Deng, "Deep Patch Visual Odometry," *Advances in Neural Information Processing Systems*, pp. 39033-39051, 2023. [Google Scholar] [Publisher Link]