STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **15**, June 2026, pp 5359–5388.
Published online in International Academic Press (www.IAPress.org)

Unpacking ORB-SLAM for UAV Navigation and Evaluating Its Efficiency Across Diverse Environments: A Systematic Review

Ahmed Mamdouh Eltaher, Ali Maher*, Ahmed. Elrewainy, Mohammed A. H. Abozied

Military Technical College, Electrical and Electronic Engineering, Cairo, Egypt

Abstract Vision-based Simultaneous Localization and Mapping (VSLAM) has become increasingly important for autonomous navigation of Unmanned Aerial Vehicles (UAVs), as it enables simultaneous localization and mapping of unknown environments using visual sensing alone. Among existing VSLAM approaches, ORB-SLAM is distinguished by its robustness under diverse operating conditions, real-time capability, and computational efficiency. This study provides a systematic and comprehensive review of the ORB-SLAM framework, with emphasis on its application in UAV navigation and mapping. We present a detailed analysis of the algorithm's core components, namely tracking, local mapping, and loop closure. In addition, the challenges associated with deploying ORB-SLAM on aerial platforms are examined, including high dynamic motion, environmental variability, and limited onboard computational resources. This work further offers a comparative evaluation of ORB-SLAM with other SLAM methodologies, highlighting its advantages and limitations across its successive variants. Finally, potential future research directions are discussed to address existing challenges, incorporate deep learning techniques, and evaluate real-world deployment feasibility. This review aims to clarify the role of ORB-SLAM in UAV applications and outline prospective development pathways.

Keywords ORB-SLAM, VSLAM Algorithm, Autonomous UAV, Diverse Environments.

DOI: 10.19139/soic-2310-5070-3442

1. Introduction

In recent times, Unmanned aerial vehicles (UAVs) have become an essential technology in many modern applications, including environmental monitoring, infrastructure inspection, search and rescue missions, and autonomous exploration. To operate autonomously in unknown environments, UAVs must continuously estimate their position while simultaneously constructing a representation of the surrounding environment. This problem is commonly referred to as Simultaneous Localization and Mapping (SLAM). Among the various SLAM paradigms, visual SLAM (VSLAM) relies primarily on camera sensors to estimate motion and reconstruct the environment using visual observations.

Visual SLAM has received significant attention in robotics and computer vision due to the low cost, lightweight hardware requirements, and rich environmental information provided by camera sensors. Compared with LiDAR-based solutions, vision-based systems can operate with lower power consumption and smaller payloads, making them particularly attractive for UAV platforms where size, weight, and energy constraints are critical. However, visual SLAM systems must operate reliably under challenging conditions such as rapid motion, illumination variations, motion blur, and scenes with limited visual texture.

Among the many visual SLAM frameworks proposed in recent years, ORB-SLAM has emerged as one of the most influential and widely adopted systems. The original ORB-SLAM algorithm introduced a keyframe-based architecture that combines efficient feature extraction, robust camera tracking, and global loop closure optimization.

*Correspondence to: Ali Maher (Email: [ali_mtc@hotmail.com]). Military Technical College, Electrical and Electronic Engineering, Cairo, Egypt.

ISSN 2310-5070 (online) ISSN 2311-004X (print)

By relying on ORB (Oriented FAST and Rotated BRIEF) features, the system achieves a favorable balance between computational efficiency and localization accuracy. Subsequent developments, including ORB-SLAM2 and ORB-SLAM3, extended the framework to support stereo cameras, RGB-D sensors, visual–inertial fusion, and multi-map management, significantly improving robustness in complex environments.

Despite these advances, deploying ORB-SLAM on aerial platforms introduces several challenges that differ from those encountered in ground robotics. UAVs frequently experience rapid camera motion, vibration-induced image blur, abrupt viewpoint changes, and varying lighting conditions. These factors can negatively affect feature detection and tracking reliability. Furthermore, onboard computational resources on UAV platforms are often limited, requiring algorithms that maintain a careful balance between real-time performance and localization accuracy. As a result, understanding the architectural design, strengths, and limitations of ORB-SLAM is essential for evaluating its suitability for UAV navigation tasks.

In recent years, a large body of research has investigated improvements and extensions of ORB-SLAM, including learning-based feature extraction, visual–inertial fusion, robust loop closure strategies, and large-scale map management. However, the rapid growth of this literature makes it difficult to obtain a comprehensive understanding of the algorithmic evolution of ORB-SLAM and its practical performance across different environments. While several studies evaluate specific improvements or applications, fewer works provide a systematic synthesis of the architectural design decisions, performance trade-offs, and empirical behavior of the ORB-SLAM family.

Motivated by this need, this paper presents a systematic review and technical analysis of the ORB-SLAM framework with particular emphasis on its applicability to UAV navigation. The study examines the evolution of ORB-SLAM and analyzes its core architectural components, including tracking, local mapping, and loop closing. In addition, the review investigates the advantages, limitations, and design trade-offs associated with feature-based SLAM systems, and compares ORB-SLAM with alternative visual SLAM approaches.

To complement the literature analysis, representative experiments using ORB-SLAM3 are conducted on widely used benchmark datasets, including KITTI and TUM RGB-D. These experiments illustrate the practical behavior of the system under different environmental conditions and highlight characteristic strengths and failure modes observed in real-world scenarios.

The remainder of this paper is organized as follows. Section 2 describes the literature search methodology used in this systematic review. Section 3 reviews the development of visual SLAM systems and the evolution of the ORB-SLAM family. Section 4 analyzes the architecture and design principles of ORB-SLAM in detail. Section 5 discusses ORB-SLAM variants, their advantages and limitations, and future research directions for improving visual SLAM systems in UAV applications.

From a practical deployment perspective, visual SLAM systems used on UAV platforms must also interact with onboard flight control systems and embedded computing hardware. In many aerial robotics platforms, algorithms such as ORB-SLAM operate alongside flight controllers including PX4 and ArduPilot, where localization accuracy directly influences navigation stability and control performance. Furthermore, UAV platforms often rely on embedded processors such as NVIDIA Jetson or ARM-based systems, which impose strict computational and energy constraints. These limitations make efficient feature-based methods particularly attractive for aerial robotics applications. At the same time, modern UAV systems increasingly incorporate visual–inertial SLAM approaches that fuse camera observations with inertial measurements to improve robustness during aggressive flight maneuvers, motion blur, or temporary visual degradation.

2. Literature Search Methodology

To ensure a rigorous and reproducible analysis of the ORB-SLAM research landscape, a systematic literature review methodology was adopted. The review process followed structured guidelines commonly used in systematic reviews in robotics and computer vision research. The methodology consisted of four main stages: defining the search strategy, identifying relevant databases, applying inclusion and exclusion criteria, and performing a structured study selection process.

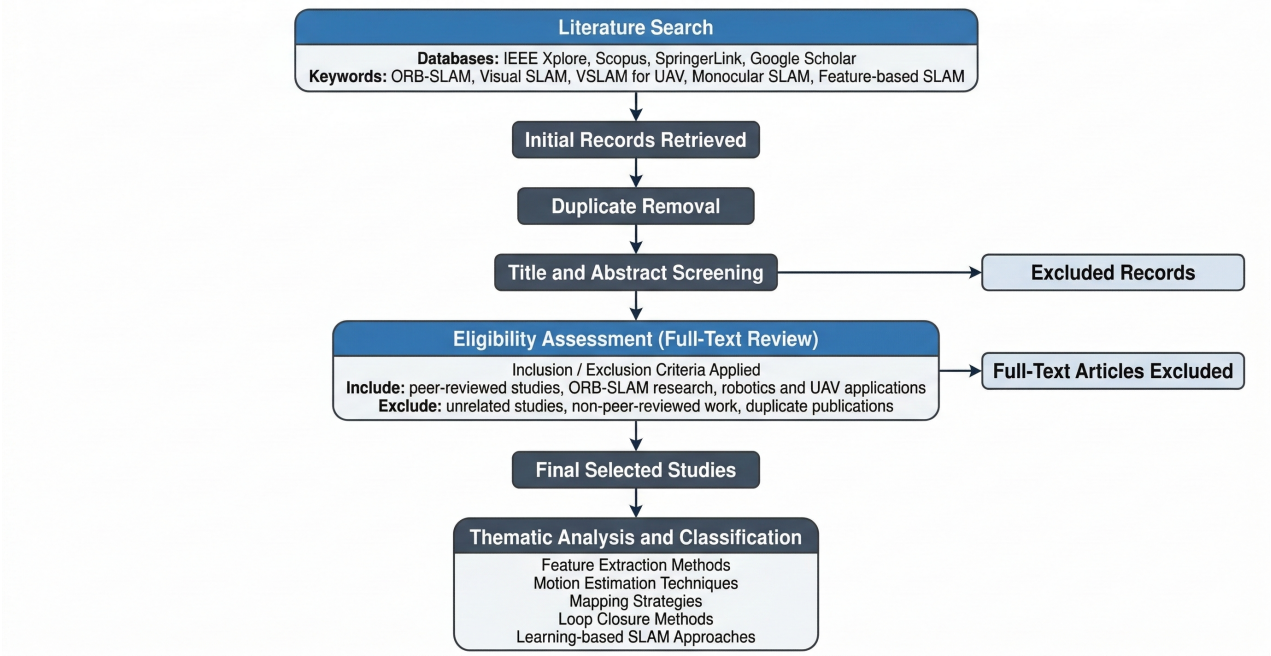

Figure 1. Workflow of the systematic literature review process used in this study. The diagram illustrates the stages of database search, duplicate removal, title and abstract screening, full-text eligibility assessment, and final study selection, followed by thematic classification of the selected ORB-SLAM research.

The study selection process was performed in multiple stages to ensure the relevance and quality of the selected literature. First, the initial search results obtained from the selected databases were screened to remove duplicate records. Next, titles and abstracts were examined to identify studies directly related to ORB-SLAM and visual SLAM systems. Publications that satisfied the inclusion criteria were then subjected to full-text analysis to confirm their relevance to UAV navigation and visual SLAM research. The final set of selected studies was analyzed to identify key algorithmic developments, architectural design choices, and practical applications of ORB-SLAM systems. The selected literature was subsequently categorized into thematic groups, including feature extraction methods, motion estimation strategies, mapping techniques, loop closure mechanisms, and learning-based SLAM approaches. This structured analysis enabled a comprehensive synthesis of the evolution and performance characteristics of the ORB-SLAM family.

3. Related work

3.1. ORB-SLAM: Evolution and Architecture

The development of ORB-SLAM has played a major role in advancing feature-based visual simultaneous localization and mapping (VSLAM) systems. Early visual SLAM research focused primarily on monocular camera systems capable of performing real-time localization and sparse map reconstruction. One of the earliest successful systems was Parallel Tracking and Mapping (PTAM), which introduced the concept of separating tracking and mapping into parallel threads, enabling real-time operation on standard hardware [1]. PTAM demonstrated that accurate camera tracking and map estimation could be achieved using keyframe-based optimization and feature correspondences.

Building upon these ideas, Mur-Artal et al. proposed ORB-SLAM, a complete monocular SLAM system that integrates tracking, local mapping, and loop closing into a unified architecture [2]. ORB-SLAM introduced several important improvements, including the use of ORB features for efficient feature extraction and matching,

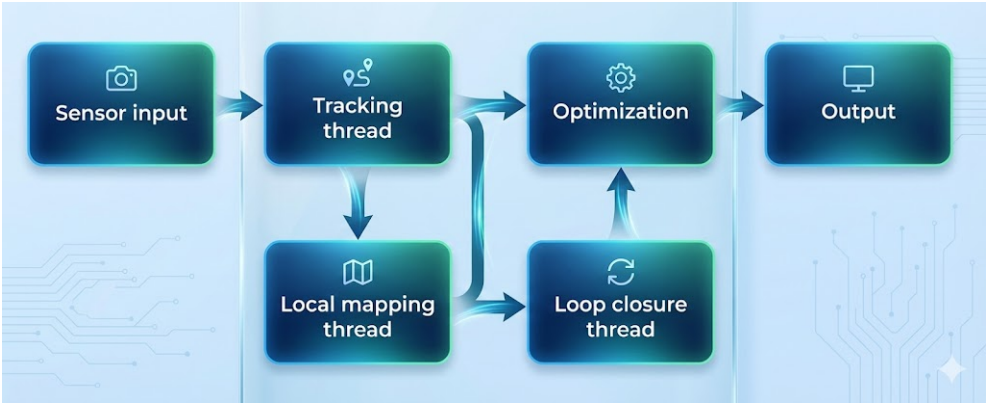

Figure 2. Illustrates the main pipelines of VSLAM algorithms.

a covisibility graph for map organization, and bundle adjustment for accurate pose estimation. The system demonstrated high accuracy and robustness across multiple datasets, establishing itself as a state-of-the-art approach for monocular SLAM.

Subsequent research extended the ORB-SLAM framework to support additional sensing modalities. ORB-SLAM2 introduced support for stereo and RGB-D cameras while maintaining the keyframe-based architecture [3]. The addition of stereo and depth sensing significantly improved scale estimation and map consistency, allowing the system to operate more reliably in large-scale environments.

More recently, ORB-SLAM3 introduced visual–inertial integration and multi-map capabilities, enabling robust localization across long-term environments and challenging motion conditions [4]. ORB-SLAM3 combines visual observations with inertial measurements from IMUs to improve robustness in situations involving rapid motion, motion blur, or temporary visual degradation.

In parallel with the development of the ORB-SLAM family, several other visual SLAM frameworks have been proposed. Direct methods such as LSD-SLAM [5] and Direct Sparse Odometry (DSO) [6] estimate camera motion directly from pixel intensities rather than discrete feature correspondences. Semi-direct approaches such as SVO [7] combine elements of both feature-based and direct methods to achieve improved computational efficiency.

Visual–inertial SLAM systems have also attracted considerable attention in recent years. Methods such as VINS-Mono [8], OKVIS [9], and OpenVINS [10] combine visual observations with inertial measurements to improve localization accuracy and robustness in dynamic environments.

Overall, the ORB-SLAM family remains one of the most widely used feature-based SLAM frameworks due to its modular architecture, computational efficiency, and strong real-time performance. Its continued development and widespread adoption highlight the importance of robust feature-based SLAM systems in robotics, autonomous vehicles, and UAV navigation.

3.2. Core Components of VSLAM Systems

Despite the diversity of visual SLAM systems, most pipelines can be decomposed into a small set of common functional modules, namely visual front-end processing, motion estimation, mapping, loop closure detection, and global or local optimization [11, 12, 13]. These components operate jointly to estimate camera motion while simultaneously constructing a representation of the surrounding environment.

The visual front-end is responsible for extracting image information that can be used for geometric inference. Depending on the algorithmic formulation, this stage may rely on handcrafted local features such as SIFT [14], SURF [15], and ORB [16], or may instead use direct image alignment methods based on pixel intensities, as in DTAM [17], LSD-SLAM [5], and DSO [6]. More recent systems increasingly incorporate learned feature representations such as SuperPoint [18], LF-Net [19], and LoFTR [20] to improve robustness in challenging environments.

The motion estimation module determines the relative camera pose between consecutive frames or keyframes. In feature-based SLAM, this is commonly achieved using epipolar geometry, PnP solvers, and nonlinear pose refinement [21, 22, 2]. In contrast, direct and semi-direct systems estimate motion through photometric consistency, as in SVO [7], LSD-SLAM [5], and DSO [6]. Visual–inertial systems extend this stage by integrating IMU measurements to improve motion estimation under rapid motion and poor visual conditions [9, 8, 4].

The mapping module maintains a geometric representation of the scene. Sparse methods such as PTAM [1] and ORB-SLAM [2] reconstruct environments using triangulated landmarks, while semi-dense and dense approaches such as LSD-SLAM [5], DTAM [17], and ElasticFusion [23] generate richer reconstructions at higher computational cost. Some systems further employ topological or hybrid representations to improve scalability and long-term consistency [24, 25].

Loop closure detection is a key component for reducing long-term drift by recognizing previously visited locations and enforcing global consistency. Classical approaches rely on bag-of-words image retrieval and geometric verification, as in FAB-MAP [24] and DBoW2 [26]. More recent methods use CNN-based global descriptors, including NetVLAD [27], to improve place recognition robustness in large-scale or appearance-changing environments.

Finally, optimization back-ends refine estimated trajectories and map structure. Bundle adjustment remains one of the most widely used techniques for jointly optimizing poses and landmarks [28], while pose graph optimization is commonly used after loop closure to distribute drift corrections efficiently [29, 30]. Together, these modules define the computational backbone of modern VSLAM systems and provide the context in which ORB-SLAM and its successors can be understood.

3.3. Feature Extraction and Matching in VSLAM

Feature extraction and matching are fundamental components of feature-based visual SLAM systems, as they provide the correspondences required for motion estimation and map construction. Robust feature detection enables reliable tracking across frames, even under viewpoint changes, illumination variation, and image noise [14, 15, 16].

Early feature-based SLAM systems relied heavily on scale- and rotation-invariant local features such as SIFT [14] and SURF [15]. These descriptors provide strong invariance properties and reliable matching performance but are computationally expensive for real-time robotic applications. To address this limitation, binary descriptors were introduced to reduce computational complexity while maintaining acceptable matching accuracy.

ORB (Oriented FAST and Rotated BRIEF) features, introduced by Rublee et al. [16], combine the FAST corner detector with the BRIEF descriptor and include orientation compensation to achieve rotation invariance. ORB features are used in ORB-SLAM due to their computational efficiency and suitability for real-time applications. Binary descriptors allow extremely fast matching using the Hamming distance, which significantly reduces computational overhead compared with floating-point descriptors such as SIFT or SURF. However, this efficiency introduces an important trade-off. Binary descriptors are generally less discriminative than gradient-based descriptors, making feature matching more susceptible to errors in highly repetitive environments or under severe illumination changes. Consequently, while ORB features enable real-time performance on resource-constrained robotic platforms, they may reduce robustness in visually ambiguous scenes. [2].

Other binary descriptors have also been proposed to improve performance under different imaging conditions. Examples include BRISK [31], FREAK [32], and AKAZE [33]. These descriptors aim to provide improved scale invariance, robustness to illumination changes, and efficient computation suitable for real-time applications.

More recently, learning-based feature extraction methods have gained significant attention in the computer vision community. Deep learning approaches such as SuperPoint [18], LF-Net [19], D2-Net [34], and R2D2 [35] learn both keypoint detection and descriptor representation directly from data. These learned representations have demonstrated improved repeatability and robustness in challenging environments compared to handcrafted features.

Transformer-based matching methods have also emerged as powerful alternatives to traditional feature matching pipelines. Methods such as LoFTR [20], SuperGlue [36], and lightglue [37] leverage attention mechanisms to establish dense correspondences between images without relying solely on local descriptors. These approaches

have shown promising results for difficult matching scenarios such as low-texture scenes and large viewpoint changes.

Despite the emergence of learning-based methods, handcrafted binary features remain widely used in real-time SLAM systems due to their computational efficiency and low memory requirements. As a result, ORB-SLAM and its successors continue to rely primarily on ORB descriptors for feature extraction and matching while maintaining compatibility with real-time robotic platforms.

3.4. Motion Estimation in Visual SLAM

Motion estimation is a fundamental component of visual SLAM systems, responsible for determining the camera pose relative to the surrounding environment. Accurate motion estimation enables reliable trajectory reconstruction and supports consistent map generation across successive frames. Most visual SLAM systems estimate the camera pose by exploiting geometric relationships between image observations and three-dimensional landmarks [21, 22].

In feature-based SLAM systems, motion estimation is typically formulated as a Perspective-n-Point (PnP) problem, where the objective is to compute the camera pose from a set of correspondences between two-dimensional image keypoints and three-dimensional map points. Efficient solutions such as EPnP [22] allow pose estimation with a minimal number of correspondences while maintaining computational efficiency suitable for real-time systems.

To improve robustness against incorrect feature correspondences, robust estimation techniques such as Random Sample Consensus (RANSAC) are commonly employed [38]. RANSAC iteratively estimates pose hypotheses using random subsets of correspondences and selects the model that maximizes the number of inliers. This strategy effectively mitigates the influence of outliers caused by mismatched features.

After obtaining an initial pose estimate, nonlinear optimization techniques are typically applied to refine the camera pose. Bundle adjustment methods minimize the reprojection error between observed image features and their corresponding projected map points [28]. Optimization algorithms such as Gauss–Newton and Levenberg–Marquardt are widely used to solve this nonlinear least-squares problem efficiently.

Several visual odometry frameworks have been proposed to improve motion estimation accuracy and computational performance. Methods such as SVO [7], Direct Sparse Odometry (DSO) [6], and ORB-SLAM [2] adopt different strategies for pose estimation depending on whether the system relies on feature-based, semi-direct, or direct image alignment approaches.

Visual-inertial fusion has also been widely investigated to improve motion estimation reliability. Systems such as OKVIS [9], VINS-Mono [8], and ORB-SLAM3 [4] integrate inertial measurements with visual observations to provide improved robustness under rapid motion, motion blur, and temporary visual degradation.

Overall, accurate motion estimation remains a central challenge in visual SLAM research, as errors in pose estimation directly affect map quality and long-term trajectory consistency. Continued research in robust optimization, multi-sensor fusion, and learning-based pose estimation aims to further improve the reliability of visual SLAM systems in complex real-world environments.

3.5. Mapping and Map Representation in SLAM

Mapping is a fundamental component of visual SLAM systems, responsible for constructing a spatial representation of the environment that supports localization and long-term navigation. In feature-based SLAM systems, the map typically consists of a collection of three-dimensional landmarks associated with visual descriptors and keyframe observations [39, 2].

Early SLAM systems such as MonoSLAM [39] used probabilistic filtering techniques to incrementally estimate both camera poses and landmark positions. However, filtering-based approaches often suffer from scalability limitations due to the growing state vector. To address this issue, modern SLAM systems adopt keyframe-based mapping strategies that decouple tracking and mapping processes.

Keyframe-based mapping was popularized by PTAM [1], which introduced parallel threads for camera tracking and map optimization. This architecture enables real-time performance while maintaining accurate map reconstruction. ORB-SLAM extends this approach by maintaining a covisibility graph that organizes relationships between keyframes based on shared map points [2].

Sparse mapping approaches reconstruct the environment using a set of triangulated landmarks derived from feature correspondences. While computationally efficient, sparse maps provide limited geometric detail. To address this limitation, semi-dense and dense SLAM methods have been proposed. Systems such as DTAM [17], LSD-SLAM [5], and ElasticFusion [23] generate richer scene representations by exploiting photometric information from image intensities.

Another important challenge in SLAM mapping is scalability in large environments. Several approaches have been proposed to improve long-term mapping performance, including submap-based representations [40] and multi-map frameworks that allow independent map segments to be created and later merged [4].

Overall, the mapping module plays a crucial role in maintaining a consistent representation of the environment while supporting accurate localization and loop closure. Improvements in map representation and optimization continue to enhance the robustness and scalability of modern SLAM systems.

3.6. Loop Closure Detection in Visual SLAM

Loop closure detection is a critical component of visual SLAM systems, as it allows the system to recognize previously visited locations and correct accumulated drift in the estimated trajectory. Without loop closure mechanisms, small pose estimation errors gradually accumulate over time, leading to significant inconsistencies in the reconstructed map [24].

Most classical loop closure systems rely on appearance-based place recognition methods. One of the earliest influential approaches is FAB-MAP [24], which uses probabilistic modeling of visual words to detect previously visited locations under varying viewpoints and illumination conditions. This method demonstrated that robust place recognition can be achieved even in large-scale environments.

Building upon these ideas, bag-of-words representations became widely adopted in visual SLAM systems. DBoW2 [26] introduced an efficient hierarchical visual vocabulary that allows fast image retrieval using binary descriptors such as ORB. This approach is used directly in ORB-SLAM for loop detection and relocalization.

Loop closure detection typically involves two stages. First, candidate loop locations are identified through appearance-based image retrieval using visual descriptors. Second, geometric verification is performed to ensure spatial consistency between the current frame and the candidate keyframe [2]. If a valid loop is detected, a loop constraint is added to the pose graph.

Recent research has explored deep learning approaches for place recognition. Methods such as NetVLAD [27], DELF [41], and AP-GeM [42] learn global image descriptors capable of recognizing places under significant viewpoint and illumination changes.

The integration of robust loop closure detection significantly improves long-term localization accuracy by enabling global map corrections through pose graph optimization. Consequently, loop closure remains one of the most important components in modern SLAM architectures.

3.7. Optimization Techniques in SLAM

Optimization plays a central role in modern SLAM systems by refining camera poses and map structure to achieve globally consistent reconstructions. Early SLAM systems relied primarily on filtering-based methods such as the Extended Kalman Filter (EKF), which jointly estimated camera poses and landmark positions within a single probabilistic framework [43, 12]. Although effective for small environments, EKF-based approaches often suffer from scalability limitations due to the increasing size of the state vector.

To address these challenges, modern visual SLAM systems employ optimization-based formulations that separate tracking from global map optimization. One of the most widely used techniques is bundle adjustment, which jointly optimizes camera poses and three-dimensional landmark positions by minimizing reprojection errors across multiple observations [28]. Bundle adjustment has become a standard component in many SLAM systems due to its ability to significantly improve map accuracy.

Graph-based SLAM formulations have also become increasingly popular. In this framework, camera poses are represented as nodes in a graph, while spatial constraints between poses are represented as edges [30]. When loop closures are detected, additional constraints are introduced, and the entire pose graph is optimized to distribute accumulated drift across the trajectory.

Several efficient optimization frameworks have been developed to support large-scale SLAM systems. Examples include g2o [29], which provides a general framework for nonlinear graph optimization, and Ceres Solver [44], which offers efficient nonlinear least-squares optimization widely used in computer vision and robotics.

Optimization techniques continue to evolve as SLAM systems become more complex. Recent work explores incremental optimization strategies and distributed optimization frameworks that improve scalability for long-term autonomous navigation in large environments.

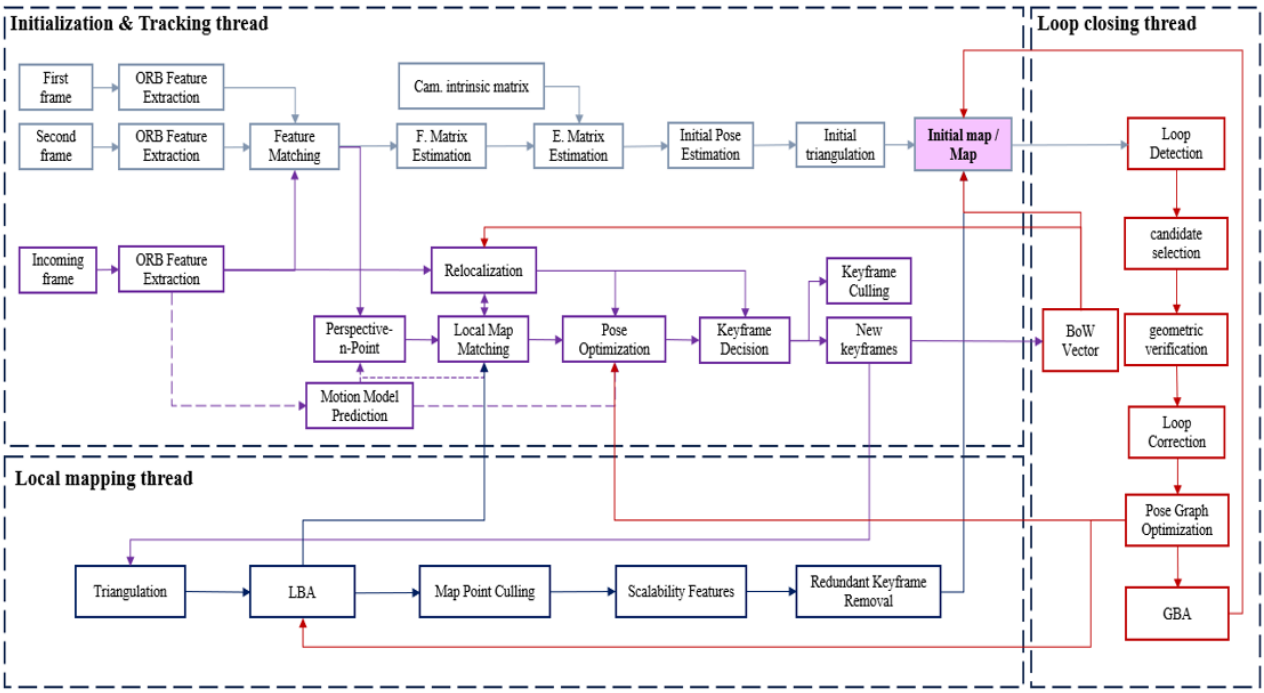

Figure 3. Depth in vision SLAM Block diagram

4. ORB-SLAM Architecture and Design Analysis

Visual Simultaneous Localization and Mapping (VSLAM) enables a robotic platform to simultaneously estimate its pose while constructing a map of the surrounding environment. Among the various VSLAM approaches, ORB-SLAM has emerged as one of the most influential frameworks due to its strong balance between computational efficiency, accuracy, and real-time performance. The system relies on ORB (Oriented FAST and Rotated BRIEF) features, which provide a lightweight alternative to classical descriptors such as SIFT and SURF while enabling fast feature extraction and matching.

The architecture of ORB-SLAM is based on three parallel threads: tracking, local mapping, and loop closing. These threads operate concurrently to estimate the camera pose, maintain a consistent map, and correct accumulated drift during long-term operation. Figure 2 illustrates the interaction between these components and highlights how information flows between the different modules of the system.

From a design perspective, ORB-SLAM adopts a sparse, feature-based formulation. This design provides significant computational advantages compared with dense or direct SLAM approaches, allowing the system to operate in real time on resource-constrained platforms such as unmanned aerial vehicles (UAVs). However, this choice also introduces several trade-offs. While feature-based methods provide strong geometric constraints in textured environments, they may struggle in scenes with limited visual texture or significant illumination changes.

Understanding these design trade-offs is therefore essential for evaluating the suitability of ORB-SLAM for different robotic applications.

The following subsections analyze the main components of the ORB-SLAM architecture. Instead of focusing on detailed mathematical derivations, the discussion emphasizes the algorithmic structure of the system, the reasoning behind key design decisions, and the practical implications of these choices in real-world environments.

4.1. Tracking thread

The tracking thread is the core component responsible for estimating the camera pose in real time. For each incoming frame, the system extracts ORB features and establishes correspondences with previously observed map points. Using these correspondences, the camera pose is estimated through geometric constraints and refined through nonlinear optimization. This process enables continuous localization of the camera while simultaneously providing observations that support map expansion in the local mapping thread.

The tracking module performs several key tasks, including feature extraction, pose estimation, and keyframe decision. These operations are tightly integrated to ensure robust camera localization while maintaining computational efficiency suitable for real-time robotic applications. The design of this module reflects an important trade-off in feature-based SLAM systems. By relying on sparse feature correspondences, ORB-SLAM significantly reduces computational complexity compared with dense or direct SLAM approaches. However, the success of this strategy depends heavily on the availability of reliable visual features in the environment.

In textured environments, feature-based tracking provides strong geometric constraints that allow accurate pose estimation. In contrast, environments with limited visual structure, motion blur, or rapid illumination changes may reduce the number of reliable feature correspondences, potentially leading to tracking degradation. Understanding these limitations is important when evaluating the suitability of ORB-SLAM for different robotic platforms, particularly UAV systems operating in challenging environments.

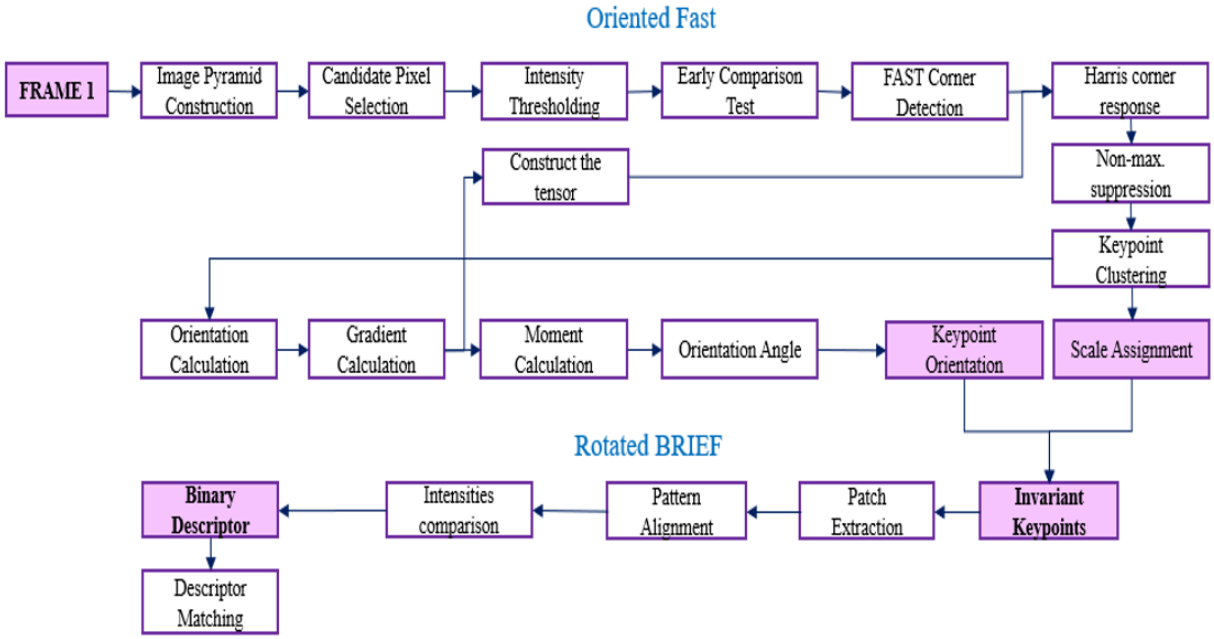

Figure 4. Block diagram illustrating the integrated threads of the ORB-SLAM algorithm

4.1.1. Feature Extraction: In the first stage of ORB-SLAM, initially, one frame is processed to extract invariant keypoints and descriptors. There are numerous operations involved as shown in Figure 3, and computation of descriptors, and these operations have been discussed sequentially below:

Feature extraction is the first stage of the tracking pipeline and plays a fundamental role in the overall performance of ORB-SLAM. The system relies on ORB (Oriented FAST and Rotated BRIEF) features to detect and describe salient points in each incoming image. As illustrated in Figure 3, the feature extraction pipeline includes multi-scale keypoint detection, orientation estimation, and binary descriptor computation.

ORB features were selected primarily for their computational efficiency. Compared with classical floating-point descriptors such as SIFT or SURF, ORB descriptors can be computed significantly faster and matched using simple Hamming distance operations. This efficiency enables ORB-SLAM to operate in real time on computationally constrained platforms, including embedded robotic systems and unmanned aerial vehicles.

However, this design choice introduces an important trade-off. Binary descriptors are generally less discriminative than gradient-based descriptors, which may lead to ambiguous matches in highly repetitive environments or under severe illumination variations. As a result, while ORB features provide excellent performance in many structured environments, the robustness of the system may degrade in scenes with weak texture or challenging lighting conditions.

To improve robustness against scale variations, ORB-SLAM constructs an image pyramid and detects keypoints at multiple resolution levels. Each detected keypoint is assigned an orientation to achieve rotation invariance, and a compact binary descriptor is computed for efficient matching. The resulting set of keypoints and descriptors forms the basis for feature correspondence between frames, enabling reliable motion estimation in subsequent stages of the tracking process.

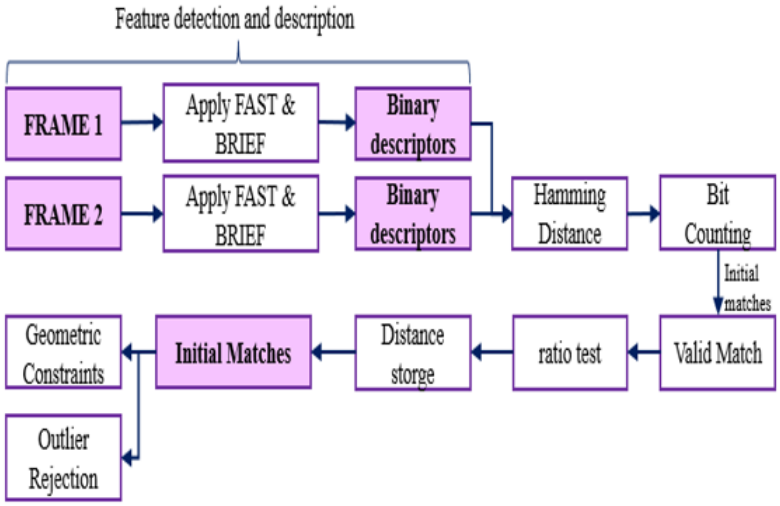

Figure 5. Block diagram represents the process of finding correspondences between two frames

4.1.2. Map Initialization: Map initialization is a critical stage in monocular ORB-SLAM because depth information cannot be directly obtained from a single camera image. The system must therefore reconstruct the initial three-dimensional structure of the scene using only two-dimensional feature observations from multiple frames.

The initialization process begins by detecting ORB features in two consecutive frames and establishing feature correspondences between them. As illustrated in Figure 4, binary descriptors are matched using Hamming distance to identify candidate correspondences. To eliminate ambiguous matches, ORB-SLAM applies robust matching strategies such as the ratio test and geometric verification.

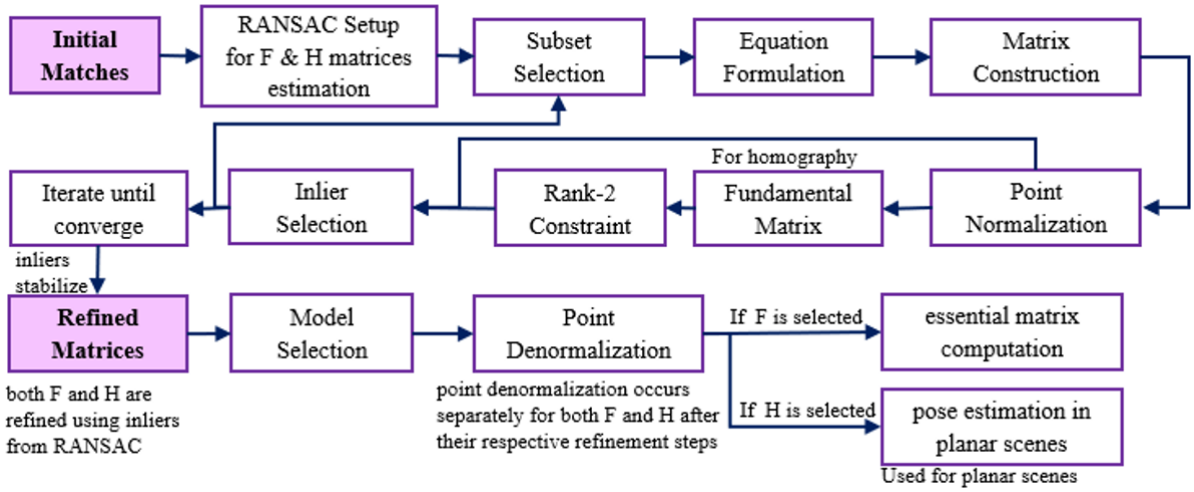

Figure 6. Illustrates the process of fundamental and homography matrices estimation

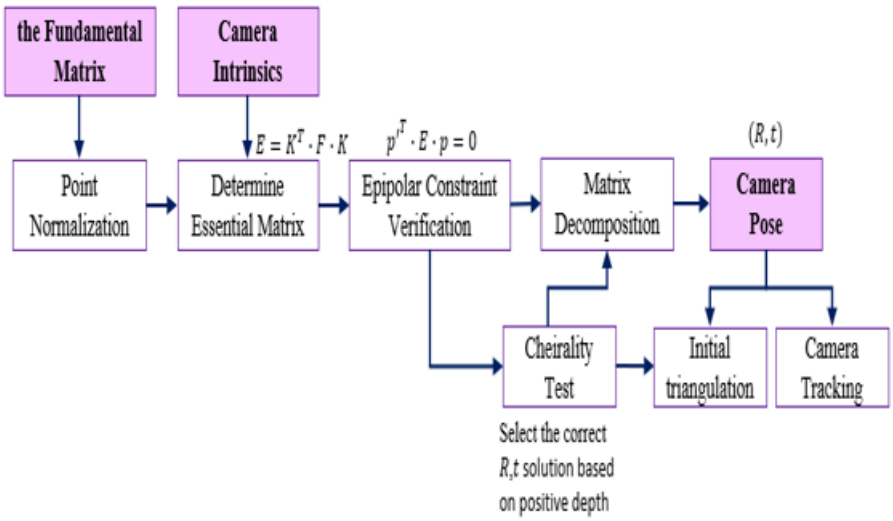

Figure 7. Initial pose estimation

Once reliable correspondences are obtained, the geometric relationship between the two frames is estimated. ORB-SLAM evaluates two possible models: a homography model, which is suitable for planar scenes, and a fundamental matrix model, which represents general camera motion in non-planar environments. As illustrated in Figure 5, these models are estimated using the RANSAC algorithm to remove outlier correspondences and obtain a robust geometric solution.

After selecting the appropriate model, the relative camera pose between the two frames is recovered. This process produces the initial estimate of camera rotation and translation, as shown conceptually in Figure 6. Using this pose estimate, the system triangulates corresponding feature points to generate an initial set of three-dimensional landmarks.

The resulting landmarks form the first sparse map of the environment and establish the reference coordinate system used throughout the SLAM process. This initialization stage plays a crucial role in determining the stability of subsequent tracking and mapping operations. A robust initialization improves pose estimation accuracy and reduces the likelihood of tracking failure during the early stages of SLAM operation.

From a design perspective, ORB-SLAM adopts a two-view geometry strategy because it offers a good balance between computational efficiency and robustness. However, monocular initialization remains inherently sensitive to scene structure and motion. For example, initialization may fail when camera motion lacks sufficient parallax or when the scene contains large textureless regions. These limitations highlight the challenges of monocular SLAM and motivate the use of additional sensors such as stereo cameras or inertial measurements in more advanced SLAM systems.

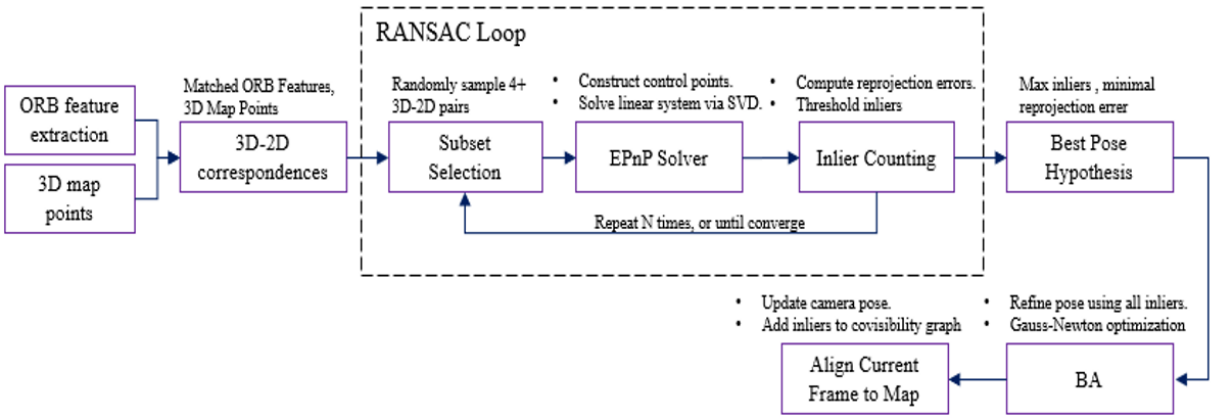

Figure 8. Illustrates the process of pose estimation

4.1.3. Motion Estimation: Motion estimation is responsible for determining the camera pose relative to the previously constructed map. This task is essential for maintaining continuous localization as the camera moves through the environment. ORB-SLAM estimates the six degrees of freedom (6-DoF) camera pose using correspondences between two-dimensional image keypoints and three-dimensional map points.

The pose estimation process typically begins by predicting an approximate camera pose based on previously estimated motion. This prediction reduces the search space for feature correspondences and improves computational efficiency. After initial correspondences between image features and map points are established, the system estimates the camera pose using the Perspective-n-Point (PnP) formulation.

Given a set of 3D map points and their corresponding 2D observations in the current frame, the camera pose is obtained by minimizing the reprojection error:

$$\min \sum_i \|x_i - \pi(TX_i)\|^2$$

where x_i represents the observed image point, X_i is the corresponding 3D map point, T denotes the camera pose transformation, and $\pi(\cdot)$ represents the camera projection function.

To improve robustness, ORB-SLAM performs nonlinear optimization using methods such as the Levenberg–Marquardt algorithm. This optimization refines the pose estimate by minimizing the reprojection error across all valid correspondences. The refined pose is then used to update the camera trajectory and determine whether the current frame should be promoted to a keyframe for map expansion.

The effectiveness of motion estimation strongly depends on the quality and distribution of feature correspondences. In environments with strong visual texture, feature-based pose estimation provides reliable geometric constraints. However, in scenes with limited texture, motion blur, or strong illumination variations, the number of reliable correspondences may decrease significantly. In such situations, alternative SLAM approaches based on direct image alignment may provide improved robustness.

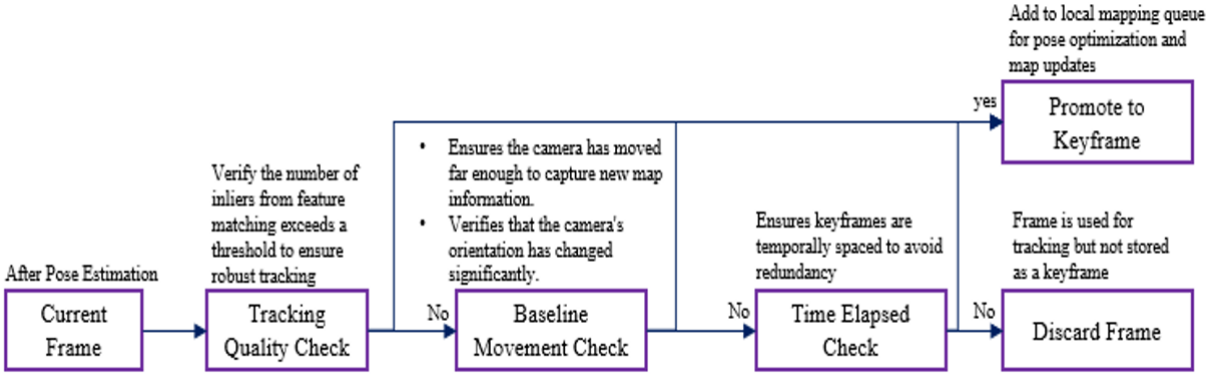

Figure 9. Illustrates the keyframe decision process

4.1.4. Keyframe Decision: Keyframe selection is an essential mechanism in ORB-SLAM that balances map accuracy and computational efficiency. Rather than inserting every frame into the map, the system selects only a subset of frames, called keyframes, that provide significant new visual information. This strategy prevents the map from growing excessively while maintaining sufficient observations for accurate localization and mapping.

As illustrated in Figure 9, several criteria are used to determine whether a frame should be promoted to a keyframe. One important factor is tracking quality. If the number of successfully tracked map points decreases significantly, inserting a new keyframe helps stabilize tracking and maintain reliable pose estimation.

Another important factor is camera motion. When the camera moves sufficiently far from the previous keyframe, new viewpoints become available for triangulating additional map points. This geometric baseline improves depth estimation accuracy and enhances map completeness.

The system also considers the presence of previously unobserved features. Frames that contain a large number of unmatched keypoints are good candidates for keyframes because they enable the creation of new map points. By selectively inserting such frames, ORB-SLAM ensures that the map gradually expands while preserving computational efficiency.

This keyframe selection strategy reflects a trade-off between map density and computational cost. Frequent keyframe insertion can improve map accuracy but increases optimization complexity, whereas sparse keyframe selection reduces computational load but may limit map completeness. ORB-SLAM therefore adopts adaptive criteria to maintain a balanced keyframe distribution during operation.

4.1.5. Keyframe Insertion: Once a frame is selected as a keyframe, it is inserted into the SLAM map and forwarded to the local mapping thread for further processing. Keyframe insertion represents the transition point between real-time tracking and map refinement. The newly created keyframe stores the estimated camera pose, the detected ORB keypoints, and the associated feature descriptors.

As illustrated in Figure 10, the insertion process also establishes connections between the new keyframe and existing keyframes through shared map points. These connections form part of the covisibility graph, which represents the structural relationships between keyframes in the map. The covisibility graph enables efficient identification of neighboring keyframes that observe common landmarks.

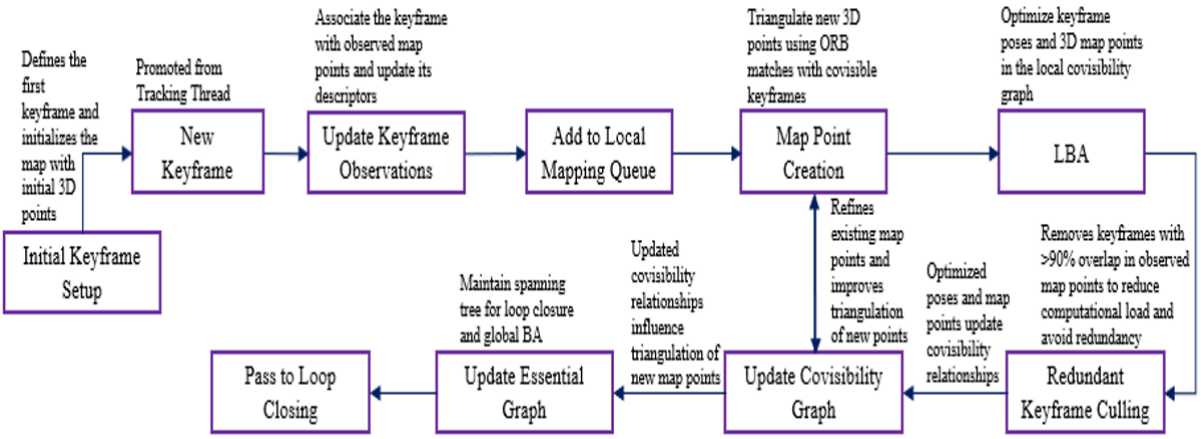

Figure 10. Block diagram illustrates the keyframe insertion

After insertion, the keyframe provides new observations that allow the system to triangulate additional map points and refine the local map structure. These operations are handled by the local mapping thread, which performs tasks such as triangulation, local bundle adjustment, and map point validation.

From a system design perspective, the separation between tracking and local mapping allows ORB-SLAM to maintain real-time performance while simultaneously improving map quality. The tracking thread focuses on fast pose estimation, whereas the local mapping thread performs more computationally intensive optimization processes asynchronously.

4.2. Local Mapping Thread

The local mapping thread is responsible for maintaining and refining the map structure generated during tracking. While the tracking thread focuses on real-time camera pose estimation, the local mapping thread performs more computationally intensive tasks that improve map accuracy and consistency. These tasks include triangulating new map points, optimizing the local map structure, and removing redundant information that may degrade system efficiency.

Local mapping operates asynchronously with respect to the tracking thread. When a new keyframe is inserted into the system, it is passed to the local mapping module, which integrates the new observations into the existing map. This separation between tracking and mapping allows ORB-SLAM to maintain real-time performance while still performing complex optimization operations in the background.

Several important operations are performed in the local mapping thread. These include updating the covisibility graph, creating new map points through triangulation, refining camera poses and map points through local bundle adjustment, and removing redundant keyframes or unreliable landmarks. Together, these operations ensure that the map remains accurate, compact, and suitable for long-term localization.

4.2.1. Covisibility Graph and Keyframe Relationships: The covisibility graph is a fundamental data structure used in ORB-SLAM to represent the spatial relationships between keyframes. In this graph, each node corresponds to a keyframe, while edges represent shared observations of common map points between keyframes. The weight of each edge reflects the number of map points jointly observed by the connected keyframes.

As illustrated in Figure 11, the covisibility graph enables the system to efficiently identify neighboring keyframes that observe similar regions of the environment. This structure plays a critical role in several SLAM operations, including map point triangulation, local bundle adjustment, and loop closure detection.

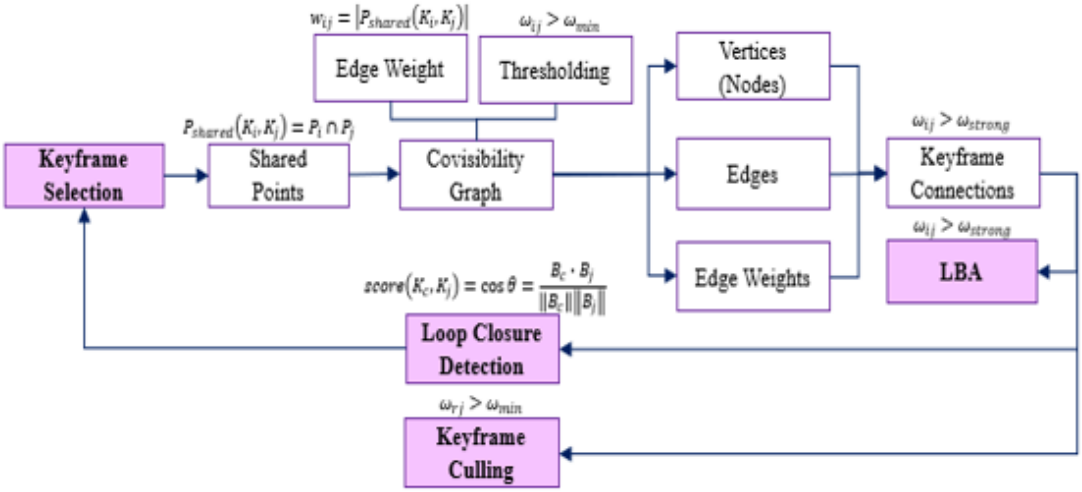

Figure 11. Illustrates the covisibility graph construction and update

When a new keyframe is inserted into the system, connections are established between the new keyframe and existing keyframes that share a sufficient number of map points. These connections allow the SLAM system to maintain a consistent representation of spatial relationships between different viewpoints.

From a system design perspective, the covisibility graph provides an efficient way to organize the map structure without requiring global connectivity between all keyframes. By maintaining only the strongest relationships between keyframes, ORB-SLAM reduces computational complexity while preserving the structural information necessary for local optimization and loop closure operations.

4.2.2. Map Point Creation (Triangulation): Triangulation is the primary mechanism used by ORB-SLAM to generate new three-dimensional landmarks from multiple image observations. When a new keyframe is inserted into the system, the local mapping thread identifies neighboring keyframes with overlapping fields of view using the covisibility graph. These neighboring keyframes provide multiple observations of the same scene features, enabling the reconstruction of their spatial positions.

The triangulation process begins by establishing feature correspondences between the newly inserted keyframe and its neighboring keyframes. These correspondences are constrained by epipolar geometry, which ensures that candidate matches satisfy the geometric relationship between the two camera views. Once reliable correspondences are identified, the spatial position of the corresponding scene point is estimated by intersecting the projection rays originating from the observing camera poses.

In practice, triangulation in the local mapping thread differs from the triangulation performed during real-time tracking. While the tracking thread prioritizes computational efficiency and performs triangulation using a limited number of frames, the local mapping thread emphasizes accuracy and robustness. Multiple observations across several keyframes are used to refine the estimated position of each landmark, and additional validation checks are applied to ensure geometric consistency.

Table 1 summarizes the key differences between triangulation in the tracking and local mapping threads. By performing triangulation within the local mapping module, ORB-SLAM can construct a more reliable and consistent map while maintaining real-time performance in the tracking process.

4.2.3. Local Bundle Adjustment: Local Bundle Adjustment (LBA) is a key optimization process used to refine the structure of the local map and improve pose estimation accuracy. After a new keyframe is inserted, the local

Table 1. Comparison of triangulation strategies in the tracking and local mapping threads.

Aspect	Triangulation in Tracking Thread	Triangulation in Local Mapping Thread
Primary Purpose	Real-time camera poses estimation and provisional map creation.	Refine and optimize the local map for accuracy and consistency.
Trigger	When a new frame arrives, or during initialization (e.g., monocular startup).	When a new keyframe is inserted or during local bundle adjustment.
Keyframes Used	1–2 recent frames (often the current frame and the last keyframe).	Multiple keyframes from the covisibility graph (e.g., 5–10 neighboring keyframes).
Triangulation Method	Linear methods (e.g., DLT) or essential matrix decomposition.	Robust methods (e.g., DLT + non-linear optimization) with bundle adjustment refinement.
Speed vs. Accuracy	Fast but less accurate (prioritizes real-time constraints).	Slower but more accurate (prioritizes map quality).
Temporal Scope	Immediate frames (short-term).	Local map window (medium-term).
Outlier Checks	Basic checks (e.g., reprojection error, epipolar constraints).	Rigorous checks (e.g., parallax thresholds, multi-view consistency, BA optimization).
Map Point Status	Provisional: Added temporarily and later validated by local mapping.	Permanent: Added to the global map after verification.
Role in SLAM	Supports camera tracking and initial mapping.	Enhances map accuracy and removes drift in the local region.
Dependencies	Depends on the current pose estimate and last keyframe.	Uses covisibility graphs and local bundle adjustment.
Optimization	Minimal (no BA).	Integrated into local bundle adjustment (optimizes poses and 3D points jointly).
Frequency	High (executed for every frame).	Lower (triggered only for new keyframes or during local BA).
Output Impact	Directly affects tracking stability and real-time localization.	Indirectly improves tracking via a refined map (used in future tracking loops).

mapping thread performs optimization over a limited set of keyframes and map points to reduce accumulated reprojection errors.

The optimization operates on a local window consisting of the newly inserted keyframe, its neighboring keyframes identified through the covisibility graph, and the set of map points observed by these keyframes. By

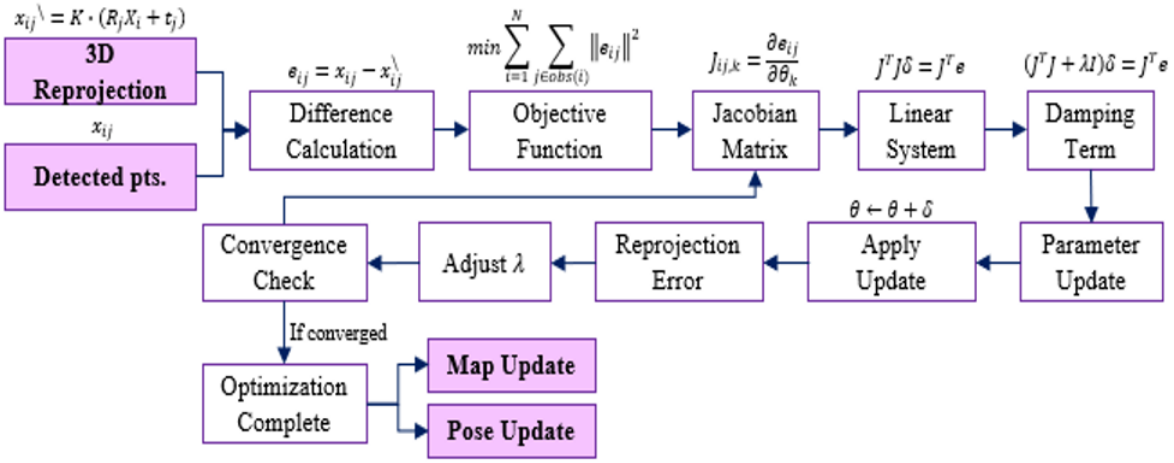

Figure 12. Illustrates the iterative process of minimizing reprojection error in LBA, updating parameters, and refining both camera poses and 3D points, as described in the textual explanation

restricting the optimization to a local region of the map, ORB-SLAM achieves a balance between computational efficiency and mapping accuracy.

The objective of local bundle adjustment is to minimize the reprojection error between observed image points and the projected positions of their corresponding three-dimensional landmarks:

$$\min_{T_i, X_j} \sum_{i,j} \rho \left(\|x_{ij} - \pi(T_i X_j)\|^2 \right)$$

where T_i represents the pose of keyframe i, X_j denotes the three-dimensional position of map point j, x_{ij} is the observed image feature, and $\pi(\cdot)$ represents the camera projection function. The robust loss function $\rho(\cdot)$ is used to reduce the influence of outliers caused by incorrect feature correspondences.

Through this optimization process, both the camera poses and the positions of map points are jointly refined. The resulting adjustments improve local map consistency and reduce drift in the estimated trajectory. Because local bundle adjustment operates only on a subset of the map, it can be executed frequently without significantly affecting real-time system performance.

4.2.4. Redundant Keyframe Culling: To maintain computational efficiency, ORB-SLAM periodically removes redundant keyframes from the map. As the system continues to explore the environment, many keyframes may capture highly similar visual information. Retaining all of these frames would unnecessarily increase the size of the map and the cost of subsequent optimization processes.

Redundant keyframes are identified by analyzing the overlap of observed map points between neighboring keyframes. If the majority of landmarks observed by a given keyframe are also observed by other keyframes with similar viewpoints, that keyframe may be considered redundant. Removing such frames reduces the number of variables involved in optimization while preserving the essential geometric structure of the map.

Once a keyframe is identified as redundant, it is removed from the map and its observations are reassigned to neighboring keyframes. The covisibility graph is then updated to reflect the new relationships between the remaining keyframes. This process helps maintain a compact map representation while ensuring that sufficient observations remain for accurate localization and mapping.

4.2.5. Map Point Culling: Map point culling is an important process that maintains the reliability and efficiency of the SLAM map by removing unstable or redundant landmarks. During operation, newly triangulated map points

may include incorrect or poorly constrained observations caused by mismatched features, insufficient parallax, or temporary tracking errors. If such points are retained, they can negatively affect pose estimation and optimization.

To maintain map quality, ORB-SLAM continuously evaluates the reliability of each map point based on several criteria. Landmarks that are observed by only a small number of keyframes or that produce large reprojection errors are considered unreliable. Similarly, points observed from nearly identical viewpoints may suffer from poor triangulation accuracy and are therefore candidates for removal.

When a map point fails these validation checks, it is removed from the map and all associated observations are discarded. This pruning process helps maintain a compact and accurate representation of the environment while preventing unreliable landmarks from degrading SLAM performance. By continuously filtering the map in this way, the system ensures that subsequent tracking and optimization processes rely on high-quality geometric information.

4.3. Loop Closing Thread

Over time, small pose estimation errors accumulate during SLAM operation, leading to drift in the estimated trajectory and inconsistencies in the reconstructed map. The loop closing thread addresses this problem by detecting previously visited locations and correcting the accumulated error through global map optimization.

Loop closing operates asynchronously with respect to the tracking and local mapping threads. When a new keyframe is created, the system attempts to determine whether the observed scene corresponds to a previously visited location. If such a loop is detected, the system estimates the geometric relationship between the current keyframe and the previously observed keyframe and applies a global correction to the map.

This process plays a critical role in maintaining long-term map consistency, particularly in large-scale environments where drift can accumulate significantly over extended trajectories. By incorporating loop closure constraints into the optimization process, ORB-SLAM is able to maintain accurate localization and mapping over long time periods.

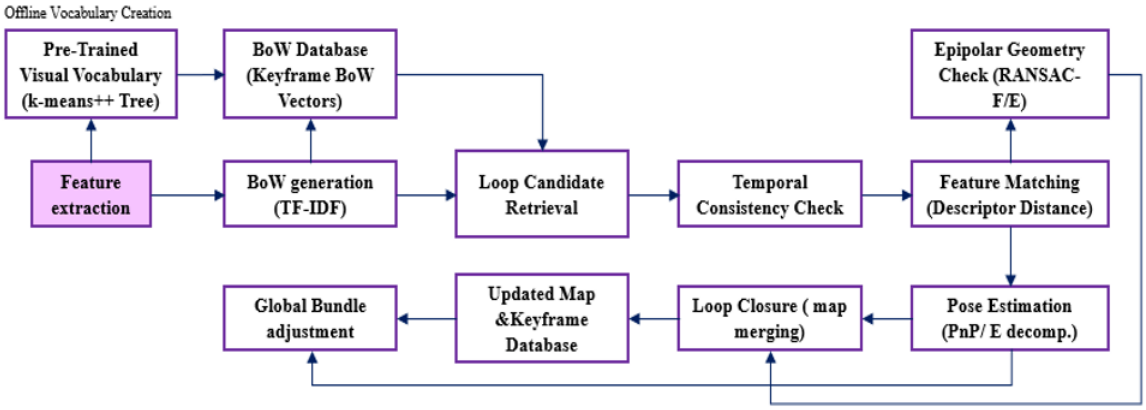

Figure 13. Illustrates the loop detection process in ORB-SLAM

4.3.1. Loop Detection: Loop detection is the first step in the loop closing process and is responsible for recognizing previously visited locations. Detecting such loops is essential for preventing long-term drift in the estimated trajectory and maintaining global map consistency.

ORB-SLAM performs loop detection using a visual place recognition technique based on the Bag-of-Words (BoW) model. In this approach, ORB feature descriptors extracted from each keyframe are quantized into visual words using a pre-trained visual vocabulary. Each keyframe is then represented by a compact BoW vector that describes the distribution of visual words observed in the scene.

When a new keyframe is created, its BoW representation is compared with those of previously stored keyframes in the database. Keyframes with similar visual word distributions are identified as potential loop candidates. As illustrated in Figure 13, this process allows the system to efficiently search for visually similar locations even in large maps.

To ensure reliability, candidate matches undergo additional geometric verification. Feature correspondences between the current keyframe and the candidate keyframe are evaluated to confirm geometric consistency between the two views. Only candidates that satisfy these geometric constraints are accepted as valid loop closures.

The use of a Bag-of-Words representation provides an efficient and scalable solution for place recognition in large-scale environments. However, this approach also introduces certain limitations. Because the vocabulary is trained offline, the system may experience reduced recognition performance in environments that differ significantly from the training data. This limitation has motivated recent research exploring learning-based place recognition methods for improved robustness.

4.3.2. Loop Correction: Once a loop has been detected and geometrically verified, the system must correct the accumulated drift that has developed in the map and the estimated camera trajectory. In monocular SLAM systems, this correction must account not only for rotation and translation errors but also for scale drift, which may arise because the absolute scale of the scene is not directly observable.

To address this issue, ORB-SLAM estimates a similarity transformation that aligns the current keyframe with the previously visited keyframe. This transformation belongs to the Sim(3) group and consists of a rotation, a translation, and a global scale factor. The estimated transformation is applied to adjust the relative poses of keyframes and the positions of associated map points so that the loop constraint is satisfied.

After computing the similarity transformation, the correction is propagated through the map structure using the keyframe connectivity relationships defined by the covisibility graph. This adjustment distributes the correction across neighboring keyframes, gradually reducing the accumulated drift in the trajectory.

The use of similarity transformations provides an effective mechanism for handling scale inconsistencies in monocular SLAM. However, the quality of the correction depends strongly on the accuracy of the detected loop and the reliability of the feature correspondences used during geometric verification.

4.3.3. Essential Graph Optimization: After loop correction is estimated, ORB-SLAM performs pose graph optimization to distribute the correction across the entire map. Instead of optimizing the full set of keyframes and landmarks immediately, the system constructs a reduced graph structure known as the essential graph.

The essential graph is a compact representation of the SLAM map that includes the most important keyframes and constraints. It contains three main types of edges: sequential edges representing the motion between consecutive keyframes, loop closure edges representing detected loop constraints, and covisibility edges representing strong spatial relationships between keyframes that observe common map points.

By optimizing this reduced graph structure, ORB-SLAM can efficiently correct large-scale trajectory drift while maintaining computational efficiency. The optimization adjusts the poses of keyframes so that both sequential motion constraints and loop closure constraints are satisfied.

This graph-based optimization approach provides an effective compromise between accuracy and computational cost. While full global optimization over all map points can be computationally expensive, optimizing the essential graph allows the system to rapidly propagate loop closure corrections across the map without interrupting real-time operation.

4.3.4. Global Bundle Adjustment: Global Bundle Adjustment (GBA) is the final optimization stage applied after a loop closure has been detected and corrected. While earlier optimization steps focus on local regions of the map or reduced graph structures, global bundle adjustment refines the entire SLAM map by jointly optimizing all keyframe poses and map point positions.

The objective of global bundle adjustment is to minimize the reprojection error across all observations in the map. By simultaneously adjusting camera poses and three-dimensional landmarks, the optimization ensures that

Table 2. Represents the key differences between essential graph optimization and global bundle adjustment

Aspect	Essential Graph Optimization (EGO)	Global Bundle Adjustment (GBA)
Objective	Maintain pose consistency efficiently by optimizing a reduced graph.	Achieve maximum accuracy by jointly optimizing all camera poses and 3D points.
Scope	Local/subset optimization (keyframes and essential edges).	Global optimization (all keyframes and 3D points in the map).
Computational Complexity	Low complexity (optimizes only keyframe poses, linear in number of keyframes).	High complexity (optimizes poses and points, scales cubically with variables).
Frequency of Execution	Frequent (e.g., during loop closure or tracking).	Infrequent (triggered after major events like loop closures or offline processing).
Parameters Optimized	Primarily camera poses (rotation and translation).	Both camera poses and 3D map point positions.
Accuracy vs. Efficiency	Prioritizes speed over absolute accuracy; trades precision for real-time performance.	Prioritizes accuracy at the cost of computational resources.
Use Cases	Real-time SLAM (e.g., robotics, AR/VR).	Offline reconstruction or high-precision applications (e.g., photogrammetry, post-processing).
Algorithm Type	Pose graph optimization (subset of nodes/edges).	Full bundle adjustment (non-linear least squares over all variables).
Dependency	Relies on covisibility graphs or prior optimizations (e.g., from local BA).	Can operate independently, often after loop closure or EGO.
Robustness	Sensitive to local errors if essential graph is incomplete.	More robust due to global consistency but requires good initialization to handle outliers.
Memory Usage	Low (stores only keyframe poses and essential edges).	High (stores all 3D points and their observations across keyframes).
Implementation Example	Used in ORB-SLAM2 for efficient loop closure correction.	Applied in ORB-SLAM2 after EGO for full-map refinement, or in COLMAP for offline SfM.

the reconstructed map is globally consistent and that the estimated trajectory accurately reflects the true motion of the camera.

Because global bundle adjustment involves a large number of variables, it is computationally expensive and therefore executed only after significant events such as loop closures. In practice, this optimization may run in a separate thread so that the tracking and mapping processes can continue operating in real time.

Although global bundle adjustment provides the highest level of accuracy, its computational cost makes it unsuitable for frequent execution in real-time systems. For this reason, ORB-SLAM typically performs faster pose graph optimization through the essential graph during normal operation and reserves global bundle adjustment for occasional global refinement steps.

4.4. Design Trade-offs and Limitations of ORB-SLAM

The architecture of ORB-SLAM reflects a series of design choices intended to balance accuracy, computational efficiency, and real-time performance [2]. By combining feature-based tracking, local map optimization, and loop closure mechanisms, the system achieves reliable localization and mapping across a wide range of environments. However, these design decisions also introduce several trade-offs that influence the performance of the system in different scenarios.

One of the key design choices in ORB-SLAM is the use of ORB features for keypoint detection and description [16]. Binary descriptors allow extremely fast feature matching using Hamming distance, enabling real-time performance even on resource-constrained platforms such as unmanned aerial vehicles (UAVs). Nevertheless, binary descriptors are generally less discriminative than floating-point descriptors such as SIFT or SURF. As a result, feature matching may become less reliable in environments with repetitive textures or significant illumination variations.

Another important design aspect is the sparse feature-based representation of the environment. Sparse mapping significantly reduces computational requirements and allows the system to operate efficiently in real time. However, this representation captures only a limited geometric description of the scene. In applications requiring dense surface reconstruction or detailed environmental modeling, dense or semi-dense SLAM methods may provide more suitable representations [5, 6, 17].

The loop closing mechanism represents one of the major strengths of ORB-SLAM. By combining visual place recognition with pose graph optimization, the system can effectively correct accumulated drift over long trajectories. The use of the Bag-of-Words model allows efficient loop detection in large-scale environments [26]. Nevertheless, the performance of this approach depends on the quality of the pre-trained visual vocabulary and may degrade in environments that differ significantly from the training data.

Finally, monocular ORB-SLAM inherently suffers from scale ambiguity because depth cannot be directly recovered from a single camera. Although loop closure and map optimization reduce drift, absolute scale remains undefined without additional sensing modalities. For this reason, many modern SLAM systems extend the ORB-SLAM framework by integrating stereo cameras, RGB-D sensors, or inertial measurements [3].

Overall, ORB-SLAM provides an effective balance between computational efficiency and mapping accuracy, which explains its widespread adoption in robotics and UAV navigation research. However, the limitations discussed above highlight important directions for future research, including improved feature representations, learning-based place recognition, and tighter integration of multi-sensor fusion techniques.

5. Variations of ORB-SLAM, Advantages, Challenges, and Future Perspectives

Particularly in UAV-based navigation, ORB-SLAM has evolved several times to satisfy the various needs of practical uses since its inception. The main deviations from the method consist of:

- The initial monocular ORB-SLAM works with one camera and is scale-ambiguous and needs other mechanisms for metric reconstruction. Later developments in ORB-SLAM2 and ORB-SLAM3 introduced multi-map handling and integration of IMUs, tremendously increasing robustness.
- A more recent adaptation known as ORB-SLAM2 uses stereo images from stereo cameras to estimate the depth. This assists in resolving scale uncertainty, resulting in monocular camera-based systems. It assists

in making UAV perform better in constrained environments in which accurate estimations of depth become paramount.

- Also unique in ORB-SLAM2, the RGB-D version uses devices like Microsoft Kinect to capture direct depth data. This version is stronger and more accurate, also for use in indoor UAV flights. Yet its use of structured light or time-of-flight depth sensing restricts its use to well-illuminated and textured surfaces.

- Through integration of multi-camera and IMU fusion, the latest update—ORB-SLAM3—improved stability in large and dynamic environments. This is particularly useful for operations by UAVs in areas without GPS. The system uses refinement of loop closure over multiple maps to gradually improve over long terms.

5.1. Advantages of ORB-SLAM Variants

The ORB-SLAM family demonstrates considerable advantages, particularly in UAV-based localization and mapping:

- ORB-SLAM depends on Oriented FAST and Rotated BRIEF (ORB) features, computationally efficient substitutes for SIFT or SURF. ORB-SLAM is fit for UAV uses since these characteristics guarantee resilience to illumination changes, perspective modifications, and modest motion blur.

- Unlike alternatives heavy in optimization (e.g., LSD-SLAM, DSO), ORB-SLAM preserves a balance between accuracy and computational efficiency, hence permitting real-time execution on UAVs with limited onboard computing capability.

- ORB-SLAM uses a Bag-of-Words (BoW) technique for loop closure detection, therefore guaranteeing consistent mapping in large-scale settings. Reusing maps considerably boosts the general applicability of the technology for long-term UAV deployment.

- Multi-Map with IMU Fusion (ORB-SLAM3): Including multi-camera and IMU fusion considerably boosts robustness, especially for high-speed UAV movements when visual features alone may be inaccurate.

5.2. Challenges Facing ORB-SLAM and Its Variants

Although ORB-SLAM has several benefits, it is not yet completely included into navigation systems using unmanned aerial vehicles (UAVs):

- Track features in a textureless environment: ORB-SLAM depends somewhat heavily on unique feature extraction. Feature tracking becomes unstable on low-textured surfaces (e.g., sky, wide fields, or tunnels), which causes tracking failures or scale drift in monocular circumstances.

- While ORB-SLAM performs well in stationary surroundings, dynamic objects—e.g., moving pedestrians, automobiles, or other UAVs—introduce misleading correspondences, hence causing erroneous loop closures and tracking failures.

- Though ORB-SLAM is tuned for real-time performance, its loop closure and posture graph optimization stage remain computationally intensive, especially for UAVs with low-power embedded processors. This calls for juggling computational economy with precision.

- Fast camera motion in UAVs could create motion blur and rolling shutter effects, therefore reducing feature tracking performance via sensitivity to camera motion and IMU noise. Furthermore, very susceptible to sensor noise and drift, IMU-based fusion in ORB-SLAM3 can cause erroneous trajectory estimates.

5.3. Empirical Behavior of ORB-SLAM Across Diverse Environments

To complement the theoretical discussion of ORB-SLAM variants, we evaluated monocular ORB-SLAM3 on two widely used benchmarks—KITTI Odometry and TUM RGB-D. These experiments reveal characteristic patterns of behavior across indoor and outdoor environments that differ significantly in motion, texture, and structural richness. The results include qualitative trajectory reconstructions and quantitative error metrics, providing a comprehensive view of system performance. Across all datasets, ORB-SLAM3 produces dense point clouds that faithfully capture local structure. However, the reconstructed trajectories reveal varying degrees of accumulated drift and several clear failure modes in more challenging scenes. Representative examples from the KITTI and TUM sequences are shown in Figures 14–24.

5.3.1. Qualitative Trajectory Analysis: To further illustrate the behavior of ORB-SLAM3 under different environmental conditions, trajectory visualizations are presented in Figures 14–23 for representative sequences from the KITTI and TUM RGB-D datasets. These figures show the estimated camera trajectories together with the reconstructed sparse maps generated during SLAM operation. The KITTI sequences demonstrate system performance in outdoor driving scenarios with varying motion patterns and structural complexity, while the TUM RGB-D sequences highlight indoor performance under conditions such as loop closure, rapid motion, and low-texture environments. Together, these qualitative results provide insight into the robustness and limitations of ORB-SLAM3 across diverse datasets.

Figures 14–18 present trajectory estimation results obtained using ORB-SLAM3 on representative sequences from the KITTI dataset, which correspond to outdoor driving scenarios with varying motion patterns and structural complexity. Figures 19–23 illustrate the corresponding results on the TUM RGB-D dataset, representing indoor environments with different challenges such as loop closure, rapid motion, and low-texture scenes. These visualizations provide qualitative insight into the behavior of ORB-SLAM3 under different environmental conditions and complement the quantitative evaluation presented in the experimental results.

5.3.2. Experimental Setup: To illustrate the practical behavior of ORB-SLAM in different environments, a set of representative experiments was conducted using publicly available benchmark datasets. The evaluation focuses on monocular ORB-SLAM3 and examines its trajectory estimation performance across both outdoor and indoor scenarios.

Two widely used datasets were selected for the analysis. The KITTI dataset was used to evaluate performance in large-scale outdoor driving environments with long trajectories and significant motion dynamics. In contrast, the TUM RGB-D dataset was used to analyze performance in structured indoor environments with shorter trajectories and different visual characteristics.

Trajectory accuracy was evaluated using two commonly adopted metrics in visual SLAM research: Absolute Trajectory Error (ATE) and Relative Pose Error (RPE). The ATE metric measures the global deviation between the estimated trajectory and the ground-truth trajectory, reflecting long-term drift accumulated during mapping. The RPE metric evaluates the local consistency of the estimated motion between consecutive frames, providing insight into short-term motion estimation accuracy.

All experiments were conducted using the publicly available ORB-SLAM3 implementation with default parameter settings. The evaluation focused on qualitative trajectory behavior and error metrics to illustrate how environmental conditions such as low texture, limited parallax, and motion dynamics influence the performance of feature-based SLAM systems.

5.3.3. Quantitative Strengths and Limitations The numerical evaluation on the KITTI and TUM datasets (Tables 3 and 4) substantiates the qualitative findings. Metrics such as the Absolute Trajectory Error (ATE), Relative Pose Error (RPE), and processing framerate provide a comprehensive assessment of system performance across diverse operational contexts.

Table 3. ORB-SLAM3 quantitative results on the KITTI odometry dataset. Absolute Trajectory Error (ATE), Relative Pose Error (RPE), mean/median tracking time, and FPS for sequences 00, 01, 02, 03, and 05.

Seq.	ATE (m)			RPE (m)			Time (ms)		FPS	difficulty grade
	RMSE	Mean	Median	RMSE	Mean	Median	Mean	Median		
00	186.61	169.61	176.38	12.73	6.78	1.24	14.81	11.98	67.5	Long urban loop; strong drift
01	38.90	34.30	39.70	18.86	11.04	2.28	11.89	11.01	84.1	Highway; high speed, low parallax
02	294.42	271.06	259.79	12.97	7.32	1.28	15.39	12.05	65.0	Very long route; extreme drift
03	153.91	134.65	136.51	10.11	6.02	1.46	13.33	12.78	75.0	Short urban; moderate drift
05	84.48	77.99	82.47	10.52	6.31	1.51	13.33	12.78	75.0	Medium loop; moderate drift

Interpretation of KITTI Results. Monocular ORB-SLAM3 exhibits high computational efficiency, sustaining 65–85 FPS across all sequences. However, it suffers from limited global consistency—a known limitation of monocular SLAM systems lacking robust loop-closure correction:

- **KITTI 00 (urban loop):** Despite a low RPE (12.7 m), ATE exceeds 186 m, indicating incomplete loop-closure correction and residual drift.
- **KITTI 01 (highway):** Low-parallax conditions lead to tracking instability and pronounced RPE spikes (up to 18.9 m).
- **KITTI 02 (longest sequence):** Accumulated scale drift results in substantial ATE ($\approx$ 294 m).
- **KITTI 03/05:** Shorter trajectories reduce cumulative drift but ATE remains significant (84–154 m), underscoring persistent scale ambiguity.

These findings corroborate established literature: monocular ORB-SLAM systems are inherently incapable of preserving metric scale over extended, unstructured paths and remain acutely sensitive to parallax, loop-closure efficacy, and scene geometry.

Table 4. ORB-SLAM3 Quantitative Results on TUM RGB-D. Performance across fr3_long_office_household, fr1_xyz, fr3_walking_xyz, fr3_nostructure_notexture_near_withloop, and fr2_large_no_loop.

Sequence	ATE RMSE (m)	ATE Mean	ATE Median	RPE RMSE	RPE Mean	RPE Median	Mean Time (ms)	Media n Time (ms)	FPS	Notes
fr3 long office	0.01728	0.01636	0.01540	0.00832	0.00731	0.00652	12.93	12.30	77.4	Stable, good performance
fr1 xyz	0.05365	0.05203	0.05158	0.01128	0.00977	0.00843	12.63	12.35	79.2	Simple motion; easy
fr3 walking	0.63569	0.61187	0.51852	0.02389	0.01853	0.01353	13.55	13.47	73.8	Very challenging (walking + blur)
fr3 nostructurewith loop	N/A	N/A	N/A	N/A	N/A	N/A	5.08	4.70	30	Immediate tracking failure
fr2 large no loop	5.001213	4.863581	5.072612	0.053473	0.015880	0.010294	11.72	11.41	30	good local consistency but noticeable global drift

Interpretation of TUM Results.

- In structured indoor environments (e.g., office and household sequences), ORB-SLAM3 achieves exceptional precision, with ATE below 2 cm.
- Performance degrades markedly under rapid motion combined with image blur (e.g., `fr3_walking_xyz`), reflecting the system's vulnerability to motion-induced visual artifacts.
- In texture-deprived or degenerate scenes (e.g., `fr3_nostructure_notexture_near`), tracking fails entirely, often precluding timestamp alignment and rendering evaluation infeasible.

- Framerate remains consistently high (70–80 FPS) under nominal conditions, but collapses immediately upon tracking loss, reflecting early system failure.

These outcomes reinforce the conclusion that ORB-SLAM3 excels in structured, textured indoor environments but is prone to catastrophic failure in low-texture or highly dynamic scenes.

Figure 14. Estimated trajectory of ORB-SLAM3 on the KITTI dataset (sequence 00). The blue curve represents the estimated camera trajectory produced by the SLAM system, while the black points correspond to the reconstructed sparse map. Red regions indicate areas where relocalization occurs during loop closure. The close overlap of the trajectory segments demonstrates consistent tracking throughout the sequence.

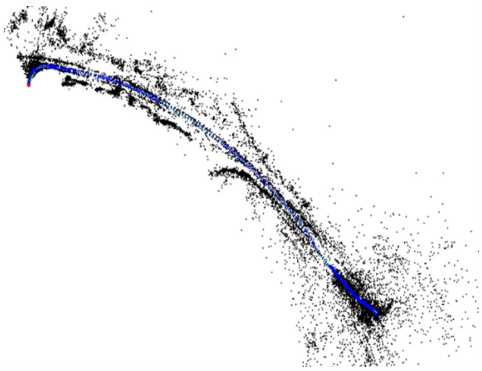

Figure 15. Estimated trajectory of ORB-SLAM3 on the KITTI dataset (sequence 01). This sequence exhibits limited lateral structure and low parallax, which weakens geometric constraints. As a result, the trajectory shows noticeable divergence despite continuous visual tracking.

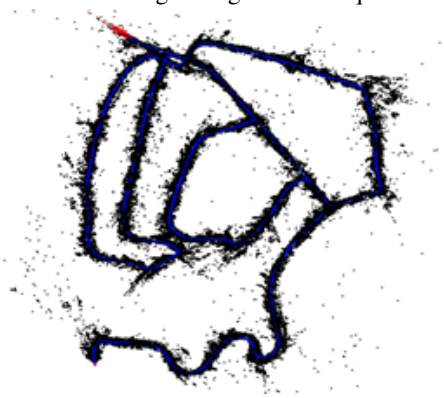

Figure 16. Estimated trajectory of ORB-SLAM3 on the KITTI dataset (sequence 02). This long and complex outdoor driving sequence illustrates the accumulation of drift over time. Red clusters near the final segment correspond to relocalization events and global map corrections triggered during loop closure.

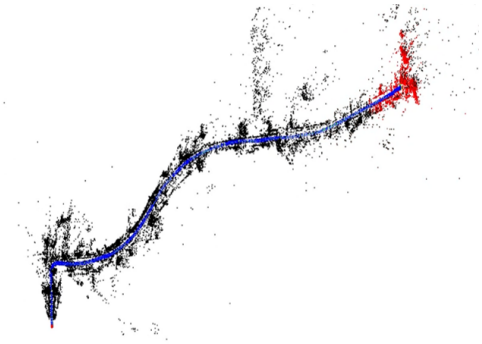

Figure 17. Estimated trajectory of ORB-SLAM3 on the KITTI dataset (sequence 03). The trajectory follows a corridor-like urban environment with relatively stable visual structure. Compared to longer sequences, the system maintains improved tracking consistency with reduced global deformation.

Figure 18. Estimated trajectory of ORB-SLAM3 on the KITTI dataset (sequence 05). This medium-length outdoor sequence demonstrates smooth tracking behavior with moderate drift. Small deviations are visible at turning points and toward the final segment of the trajectory.

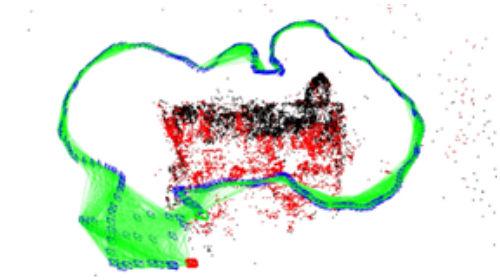

Figure 20. Estimated trajectory of ORB-SLAM3 on the TUM RGB-D dataset (long office household sequence). This long indoor trajectory includes repeated loop closures that help maintain global map consistency. The presence of rich texture and structured indoor geometry supports robust tracking and mapping.

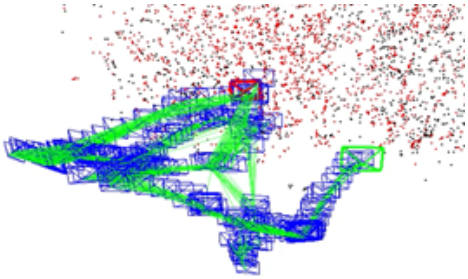

Figure 22. Estimated trajectory of ORB-SLAM3 on the TUM RGB-D dataset (walking motion sequence). Rapid camera motion and frequent viewpoint changes introduce motion blur and tracking difficulty, resulting in trajectory instability toward the final part of the sequence.

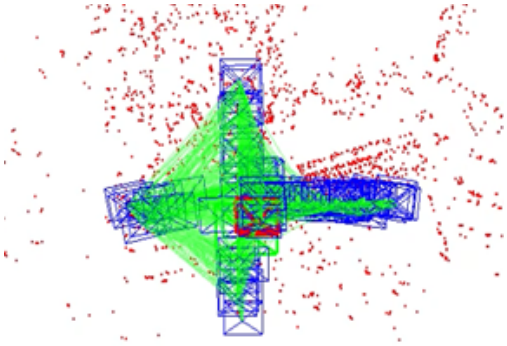

Figure 19. Estimated trajectory of ORB-SLAM3 on the TUM RGB-D dataset (sequence fr1 xyz). The indoor trajectory follows a simple and well-structured motion pattern, producing a coherent sparse map. The stable camera motion and rich visual features allow the SLAM system to maintain accurate pose estimation throughout the sequence.

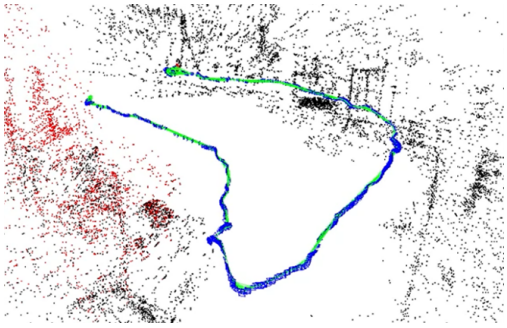

Figure 21. Estimated trajectory of ORB-SLAM3 on the TUM RGB-D dataset (large indoor sequence without loop closure). Although local tracking remains stable, the absence of loop closure constraints leads to visible global drift along the trajectory.

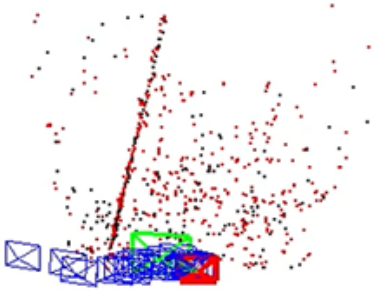

Figure 23. Estimated trajectory of ORB-SLAM3 on the TUM RGB-D dataset (low texture and low structure sequence). The lack of reliable visual features leads to rapid tracking failure, visible as a collapsed trajectory near the starting point.

5.3.4. Discussion: The qualitative reconstructions and quantitative measurements collectively offer an in-depth understanding of the operational features and constraints of monocular ORB-SLAM3 across diverse environments. Multiple consistent themes arise that illustrate the relationship between environmental structure, motion dynamics, and the algorithmic assumptions foundational to feature-based monocular SLAM.

The first thing to note is that the findings show that ORB-SLAM3 keeps great local tracking accuracy when the lighting is good. Indoor sequences with lots of texture, recurrent structural cues, and moderate camera movement, like `fr3_long_office_household` and `fr1_xyz`, show smooth paths and point clouds that are dense and mathematically cohesive. The low ATE values observed in these sequences demonstrate that the system's front-end (feature extraction, matching, and initial pose estimation) and back-end (local bundle adjustment) are highly effective in constrained and well-structured scenes. This is further reflected in the consistently high frame rates reported across all indoor datasets, emphasizing the computational efficiency that continues to distinguish ORB-SLAM from more computationally demanding learning-based approaches.

However, even in these favorable settings, the system's global consistency can vary. Sequences with higher spatial extents or less favorable loop-closure possibilities (e.g., `fr2_large_no_loop`) exhibit significant drift notwithstanding precision estimations of motion over a short time. This observation shows a common problem with monocular SLAM: global alignment over longer paths stays fundamentally ill-posed without strong loop-closure constraints, even though relative motion can be predicted very precisely. In these cases, the difference between low RPE and much higher ATE numbers makes the difference between local geometric consistency and global map correctness even clearer.

The outdoor KITTI experiments expose the second major theme: accumulated scale drift and structural deformation during long-range, high-speed motion. KITTI sequences 00, 02, and 05 demonstrate that monocular ORB-SLAM3 is highly sensitive to changes in environmental structure and available parallax. Although tracking remains continuous, the resulting trajectories deviate markedly from the underlying road geometry, especially when loop closures are missed or insufficient. KITTI 01, characterized by highway Driving with almost purely forward motion exemplifies the system's vulnerability to low-parallax conditions. The limited diversity in viewpoint restricts the ability of the SLAM system to triangulate new landmarks, causing scale ambiguity, unstable PnP estimation, and elevated pose error.

A third insight comes from the challenging TUM sequences: failure modes triggered by visual degradation or structural sparsity. The quick failure on `fr3_nostructure_notexture_near_withloop` shows how important it is for ORB feature detection to work on surfaces with textures. The front end does not produce good matches because it does not have corner-like features or unique keypoints. This causes the pose estimation pipeline to quickly diverge. In the same way, `fr3_walking_xyz`, which has a lot of motion blur and quick changes in viewpoint, shows how sensitive ORB-SLAM3 is to image quality loss. The system tries to recover, but the resulting path is unstable, and the latter part of the path shows a lot of divergence, which is supported by the much higher ATE values.

Taken together, these findings illustrate the trade-off at the core of ORB-SLAM's design: high computational efficiency and strong performance in structured, textured environments come at the cost of sensitivity to scale drift, texture scarcity, forward-motion degeneracy, and motion blur. The results also highlight a clear distinction between environments that conform well to ORB-SLAM's underlying assumptions (static, textured, and structurally stable scenes) and those that challenge feature-based monocular pipelines.

These observations collectively motivate several directions for improving robustness and generalization. Enhancing feature extraction using texture-adaptive or learned descriptors may mitigate failures in low-texture environments. Mechanisms for scale correction—such as multi-keyframe fusion, lightweight global optimization, or alternative structural priors—may alleviate long-range drift. Finally, integrating motion-aware keyframe selection, blur-robust matching strategies, and predictive modeling could improve resilience under rapid or irregular camera motion.

Overall, the empirical evidence presented across both KITTI and TUM benchmarks provides a grounded understanding of ORB-SLAM3's performance envelope and highlights the areas where future improvements can yield substantial gains in robustness and map consistency.

5.4. Future Perspectives for Addressing These Challenges

Future research has to solve ORB-SLAM's limitations by providing the following improvements so that it remains a practical option for mapping and localization with UAVs:

- **Integrating Learning-Based Feature Extraction:** Standard ORB feature extraction breaks down in areas of textureless regions. Novel improvements in learning-based feature descriptors—e.g., SuperPoint and R2D2—provide an encouraging direction by learning stronger and discriminatory features. Future ORB-SLAM algorithms can employ a hybrid feature extraction, combining handcrafted and learning-based descriptions.

- **Dynamic Scene Understanding:** In order to get rid of limitations in dynamic environments, it entails adding semantic segmentation to rule out dynamic objects in the scene. Deep-learning-based scene parsing or optical flow-based motion segmentation techniques would significantly enhance robustness.

- **Edge AI Hardware Acceleration:** The growing availability of edge AI hardware (e.g., NVIDIA Jetson and Google Coral TPU) presents the possibility for running optimal ORB-SLAM solutions capable of exploiting parallelized GPU processing. Pose graph optimization derived from GPGPU can greatly lower computational overhead.

- **Multi-Sensor Fusion and Event Cameras:** Future UAV missions can involve event cameras, which capture high-speed motion cues without motion blur. Additionally, LiDAR–visual fusion can allow more accurate localization in challenging environments such as forests or urban canyons.

- **High-Speed UAV Maneuver Robust SLAM:** Extending ORB-SLAM to enable higher frame-rate tracking and predictive motion models will make it more robust for high-speed flights of UAVs. Using reinforcement learning-based trajectory prediction can be employed for improving UAV navigation in very dynamic environments.

6. Conclusion

This paper presented a systematic review and technical analysis of the ORB-SLAM framework with a particular focus on its applicability to autonomous UAV navigation. The study examined the architectural design of ORB-SLAM, including its tracking, local mapping, and loop closing threads, and analyzed how these components interact to achieve accurate real-time visual simultaneous localization and mapping.

Through the analysis of the ORB-SLAM pipeline, this review highlighted several key factors that contribute to the success of the framework. The use of ORB features enables efficient feature extraction and matching suitable for real-time robotic systems. The keyframe-based architecture combined with local bundle adjustment provides accurate pose estimation while maintaining computational efficiency. In addition, the integration of loop closure detection and pose graph optimization allows the system to correct accumulated drift and maintain global map consistency over long trajectories.

Despite these strengths, several limitations remain. Feature-based SLAM methods such as ORB-SLAM rely heavily on the availability of reliable visual features, which can lead to performance degradation in low-texture environments, dynamic scenes, or under severe illumination changes. Furthermore, monocular configurations suffer from inherent scale ambiguity, and the computational cost of global optimization may limit performance on resource-constrained UAV platforms.

The experimental observations across KITTI and TUM datasets illustrate how ORB-SLAM behaves under different environmental conditions, revealing both its robustness in structured environments and its limitations in challenging scenarios such as low-parallax motion, dynamic scenes, or weak visual texture. These findings highlight the importance of considering environmental characteristics when deploying visual SLAM systems in UAV applications.

Future research directions include the integration of learning-based feature representations, improved place recognition methods, and tighter visual–inertial sensor fusion to enhance robustness in challenging environments. Additionally, advances in lightweight optimization techniques and hardware-aware implementations may further improve the feasibility of deploying SLAM algorithms on embedded UAV platforms.

Overall, ORB-SLAM remains one of the most influential feature-based visual SLAM frameworks due to its balance between accuracy, efficiency, and modular design. Continued research combining classical geometric methods with modern learning-based approaches is expected to further advance the capabilities of visual SLAM systems for autonomous aerial robotics.

REFERENCES

1. G. Klein and D. Murray, *Parallel Tracking and Mapping for Small AR Workspaces*, in 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, pp. 225–234, 2007.
2. R. Mur-Artal, J. M. M. Montiel, and J. D. Tard'os, *ORB-SLAM: A versatile and accurate monocular SLAM system*, IEEE Transactions on Robotics, vol. 31, no. 5, pp. 1147–1163, 2015.
3. R. Mur-Artal and J. D. Tard'os, *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.
4. C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, *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.
5. J. Engel, T. Schöps, and D. Cremers, *LSD-SLAM: Large-scale direct monocular SLAM*, in European Conference on Computer Vision, pp. 834–849, Springer, 2014.
6. J. Engel, V. Koltun, and D. Cremers, *Direct sparse odometry*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, no. 3, pp. 611–625, 2017.
7. C. Forster, M. Pizzoli, and D. Scaramuzza, *SVO: Fast semi-direct monocular visual odometry*, in 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 15–22, 2014.
8. T. Qin, P. Li, and S. Shen, *VINS-Mono: A robust and versatile monocular visual-inertial state estimator*, IEEE Transactions on Robotics, vol. 34, no. 4, pp. 1004–1020, 2018.
9. S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, *Keyframe-based Visual–Inertial Odometry using Nonlinear Optimization*, in Proceedings of Robotics: Science and Systems (RSS), 2015.
10. P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, *OpenVINS: A Research Platform for Visual-Inertial Estimation*, in IROS 2019 Workshop on Visual-Inertial Navigation: Challenges and Applications, 2020.
11. C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, *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.
12. H. Durrant-Whyte and T. Bailey, *Simultaneous localization and mapping: part I*, IEEE Robotics & Automation Magazine, vol. 13, no. 2, pp. 99–110, 2006.
13. 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.
14. D. G. Lowe, *Distinctive image features from scale-invariant keypoints*, International Journal of Computer Vision, vol. 60, no. 2, pp. 91–110, 2004.
15. H. Bay, T. Tuytelaars, and L. Van Gool, *SURF: Speeded up robust features*, in European Conference on Computer Vision, pp. 404–417, Springer, 2006.
16. E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, *ORB: An efficient alternative to SIFT or SURF*, in 2011 International Conference on Computer Vision, pp. 2564–2571, 2011.
17. R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, *DTAM: Dense tracking and mapping in real-time*, in 2011 International Conference on Computer Vision, pp. 2320–2327, 2011.
18. D. DeTone, T. Malisiewicz, and A. Rabinovich, *SuperPoint: Self-supervised interest point detection and description*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 224–236, 2018.
19. Y. Ono, E. Trulls, P. Fua, and K. M. Yi, *LF-Net: Learning local features from images*, Advances in Neural Information Processing Systems, vol. 31, 2018.
20. J. Sun, Z. Shen, Y. Wang, H. Bao, and X. Zhou, *LoFTR: Detector-free local feature matching with transformers*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8922–8931, 2021.
21. R. I. Hartley and A. Zisserman, *Multiple view geometry in computer vision*, 2nd ed. Cambridge: Cambridge University Press, 2004.
22. V. Lepetit, F. Moreno-Noguer, and P. Fua, *EP n P: An accurate O (n) solution to the P n P problem*, International Journal of Computer Vision, vol. 81, no. 2, pp. 155–166, 2009.
23. T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison, *ElasticFusion: Dense SLAM without a pose graph*, in Robotics: Science and Systems, vol. 11, no. 3, 2015.
24. M. Cummins and P. Newman, *FAB-MAP: Probabilistic localization and mapping in the space of appearance*, The International Journal of Robotics Research, vol. 27, no. 6, pp. 647–665, 2008.
25. M. Labbé and F. Michaud, *RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation*, Journal of Field Robotics, vol. 36, no. 2, pp. 416–446, 2019.
26. D. Gálvez-López and J. 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.

27. R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, *NetVLAD: CNN architecture for weakly supervised place recognition*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5297–5307, 2016.
28. B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, *Bundle adjustment—a modern synthesis*, in International Workshop on Vision Algorithms, pp. 298–372, Springer, 1999.
29. R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, *g 2 o: A general framework for graph optimization*, in 2011 IEEE International Conference on Robotics and Automation, pp. 3607–3613, 2011.
30. G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, *A Tutorial on Graph-Based SLAM*, IEEE Intelligent Transportation Systems Magazine, vol. 2, no. 4, pp. 31–43, 2010.
31. S. Leutenegger, M. Chli, and R. Y. Siegwart, *BRISK: Binary robust invariant scalable keypoints*, in 2011 International Conference on Computer Vision, pp. 2548–2555, 2011.
32. A. Alahi, R. Ortiz, and P. Vandergheynst, *FREAK: Fast retina keypoint*, in 2012 IEEE Conference on Computer Vision and Pattern Recognition, pp. 510–517, 2012.
33. P. F. Alcantarilla, J. Nuevo, and A. Bartoli, *Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces*, in Proceedings of the British Machine Vision Conference (BMVC), pp. 1–11, 2013.
34. M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler, *D2-net: A trainable cnn for joint description and detection of local features*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8092–8101, 2019.
35. J. Revaud, C. De Souza, M. Humenberger, and P. Weinzaepfel, *R2d2: Reliable and repeatable detector and descriptor*, Advances in Neural Information Processing Systems, vol. 32, 2019.
36. P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, *SuperGlue: Learning feature matching with graph neural networks*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4938–4947, 2020.
37. P. Lindenberger, P.-E. Sarlin, and M. Pollefeys, *LightGlue: Local feature matching at light speed*, in Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 17627–17638, 2023.
38. M. A. Fischler and R. 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.
39. A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, *MonoSLAM: Real-time single camera SLAM*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, no. 6, pp. 1052–1067, 2007.
40. K. Konolige and M. Agrawal, *Large-Scale Visual Odometry for Rough Terrain*, in Proceedings of the International Symposium on Robotics Research (ISRR), 2010.
41. H. Noh, A. Araujo, J. Sim, T. Weyand, and B. Han, *Large-scale image retrieval with attentive deep local features*, in Proceedings of the IEEE International Conference on Computer Vision, pp. 3456–3465, 2017.
42. J. Revaud, J. Almazán, R. S. Rezende, and C. R. de Souza, *Learning with average precision: Training image retrieval with a listwise loss*, in Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 5107–5116, 2019.
43. R. Smith, M. Self, and P. Cheeseman, *Estimating uncertain spatial relationships in robotics*, in Autonomous Robot Vehicles, pp. 167–193, Springer, 1990.
44. S. Agarwal, K. Mierle, and Others, *Ceres Solver: Tutorial and Reference*, in Google Inc., 2012.