Contents lists available at ScienceDirect

Displays

journal homepage: www.elsevier.com/locate/displa

Review

Review on SLAM algorithms for Augmented Reality☆

Xingdong Sheng [a,b], Shijie Mao [b], Yichao Yan [a], Xiaokang Yang [a,*]

[a] *The MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiaotong University, Dongchuan Road 800, Shanghai 200240, China*
[b] *Lenovo Research, Songtao Road 696, Shanghai 201210, China*

ARTICLE INFO

Keywords:
Augmented reality
SLAM
Visual SLAM
Visual-inertial

ABSTRACT

Augmented Reality (AR) has gained significant attention in recent years as a technology that enhances the user's perception and interaction with the real world by overlaying virtual objects. Simultaneous Localization and Mapping (SLAM) algorithm plays a crucial role in enabling AR applications by allowing the device to understand its position and orientation in the real world while mapping the environment. This paper first summarizes AR products and SLAM algorithms in recent years, and presents a comprehensive overview of SLAM algorithms including feature-based method, direct method, and deep learning-based method, highlighting their advantages and limitations. Then provides an in-depth exploration of classical SLAM algorithms for AR, with a focus on visual SLAM and visual-inertial SLAM. Lastly, sensor configuration, datasets, and performance evaluation for AR SLAM are also discussed. The review concludes with a summary of the current state of SLAM algorithms for AR and provides insights into future directions for research and development in this field. Overall, this review serves as a valuable resource for researchers and engineers who are interested in understanding the advancements and challenges in SLAM algorithms for AR.

Contents

☆ This paper was recommended for publication by Guangtao Zhai.

* Corresponding author.
E-mail address: xkyang@sjtu.edu.cn (X. Yang).

https://doi.org/10.1016/j.displa.2024.102806
Received 22 April 2024; Received in revised form 12 July 2024; Accepted 28 July 2024
Available online 31 July 2024

1. Introduction

In the past 10 years, there has been significant development in ARVR. Many applications of ARVR have emerged in various fields [1–4] such as gaming, education, healthcare, and architecture. These are all benefits of AR which can provide the ability to combine the real world with virtual elements to create immersive and interactive experiences. Hololens, MagicLeap, and Apple Vision Pro are instances of AR devices that enable users to view and engage with virtual objects in the physical world. Operating systems like IOS and Android offer ARKit and ARcore to assist developers in creating their own AR apps. Spatial computing is the essential technology for creating immersive and interactive experiences, often facilitated by devices or SDKs. In the context of AR, spatial computing enables a deep understanding of the physical environment, ensuring that virtual objects align and interact seamlessly with the real world. This is achieved through addressing three fundamental questions: where am I? What is around me? How do I interact with virtual objects? These questions are answered through key algorithms such as SLAM, 3D reconstruction, and gesture recognition. SLAM is a crucial component in spatial computing algorithms for AR, with 3D reconstruction and hand gesture recognition being additional features. This review paper specifically focuses on reviewing SLAM algorithms for AR.

There are many works on the review and survey of SLAM, such as [5–8], but they are all separate reviews focusing on algorithms themselves, [9] introduced the application of deep learning in visual SLAM. There are very few reviews specifically targeting SLAM in a particular application field, [10,11] discussed the SLAM algorithm for autonomous vehicles and underwater vehicles. There is very little comprehensive review work about SLAM algorithms for AR, [12,13] gives a survey of AR and pose estimation for AR in early years, [14] only introduced monocular visual-inertial SLAM algorithms for augmented reality.

This paper offers a comprehensive and systematic review and discussion of SLAM algorithms for AR. It provides an overview of the current status, methods, and challenges in these algorithms, aiming to serve as a technical and practical reference for researchers working on AR technology, particularly those focused on SLAM algorithm research and development in AR. The article also aims to provide guidance on improvement directions and principles for SLAM algorithms in AR.

The paper begins with a short introduction to SLAM for AR, providing a historical overview of classical AR products and SLAM algorithms, as well as the fundamental principles of using SLAM in AR. It then explores different SLAM techniques for AR, such as various types of visual SLAM, classical visual SLAM algorithms for AR, and visual-inertial SLAM for AR. The paper also analyzes sensor configuration, datasets, and typical algorithm performance for AR SLAM. Finally, it discusses the current status and challenges of SLAM for AR in a concluding section.

2. Background

2.1. AR products with SLAM

As shown in Fig. 1, In the past few years, a number of important AR products have become available, including Hololens, Magic Leap, ARKit, ARCore, and Apple Vision Pro. Each of these products has its own distinct features and abilities that aim to improve user experiences.

Wikitude [15], Vuforia [16], and EasyAR [17] were some of the pioneers in integrating AR SDKs with SLAM pose estimation, enabling users to enjoy AR applications on mobile and other portable devices, and these SDKs are still being continuously optimized.

Microsoft's Hololens is the first AR headset that blends the physical and digital worlds seamlessly. It features advanced sensors, cameras, and a holographic processing unit to enable real-time precise SLAM tracking. Hololens provides the first and also the best optical see-through (OST) AR glass experience. Magic Leap's AR glasses are another OST glasses that combine cutting-edge optics with spatial computing technology to deliver a mixed reality experience. The device uses light-field technology to project digital content with depth and realism, creating lifelike virtual objects in the user's environment. Oculus Quest is a cutting-edge headset with a video see-through (VST) experience that allows users to seamlessly blend virtual content with high-quality and real-time video of the real world, similar to Oculus Quest, Apple Vision Pro represents the most advanced AR device currently available, by utilizing many sensors, sophisticated algorithms, and powerful computing power, it not only achieves high-performance SLAM spatial tracking but also enables natural gesture interaction and environmental recognition, ultimately delivering an amazing AR experience.

Apple's ARKit framework brings AR capabilities to millions of iOS devices, allowing users to experience augmented reality on their iPhones and iPads. ARKit uses advanced computer vision techniques and SLAM algorithms to enable accurate tracking of surfaces, objects, and motion in the real world. Google's ARCore platform brings augmented reality to Android devices, offering similar capabilities to Apple's ARKit. ARCore uses motion tracking, environmental understanding, and light estimation to enable AR experiences on a wide range of Android smartphones and tablets.

Qualcomm's XR1/XR2 chipsets are designed to deliver high-performance capabilities for extended reality (XR) devices, including AR and VR headsets, through advanced SLAM algorithms. Leading manufacturers in the AR/VR industry, such as Lenovo and HTC, have integrated this chipset into their devices to boost performance.

2.2. Development of SLAM algorithm

In the past decade, with the development of actual product demand and new computer vision and deep learning technologies, a large number of SLAM algorithms have been proposed, as shown in Fig. 1. These algorithms aim to enhance more accurate and efficient real-time mapping and navigation in various applications, ranging from robotics and autonomous vehicles to augmented reality and virtual reality systems. Researchers and developers continue to innovate in this field, exploring new techniques and approaches to further improve the performance and reliability of SLAM systems. These proposed SLAM algorithms are aimed at addressing two basic problems which are the real-time requirements under limited computing resources and higher precision and stability performance under different scenarios.

Firstly, driven by the real-time requirements under limited computing resources, a large number of methods have been proposed to improve the real-time performance of algorithms. For example, MonoSLAM [18], PTAM [19], DTAM [20], ORB-SLAM [21,22], and many related algorithms [23–27] optimized the algorithm execution

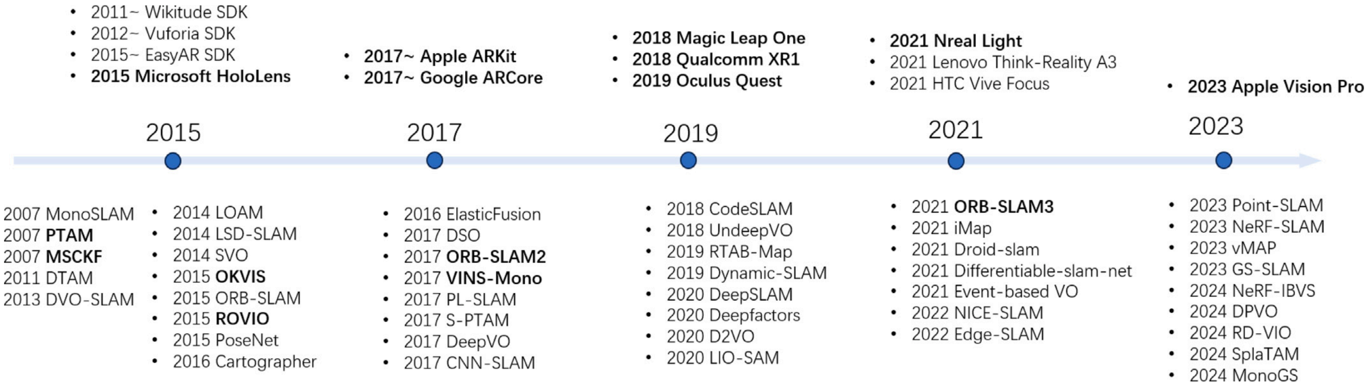

Fig. 1. AR products and SLAM algorithms in recent years.

process, such as feature extraction, matching methods and parallel processing methods to enhance real-time performance. To get higher real-time performance, a large number of IMU fusion-based SLAM algorithms were proposed to reduce the complexity of visual information processing by fusing IMU information, thereby reducing the computing power requirements and greatly increasing the output frame rate of SLAM, such as MSCKF [28], OKVIS [29], ROVIO [30], and many later related works such as [31–33]. In recent years, some works such as RD-VIO [34] optimized for the problems of dynamic scenes and pure rotation by integrating IMU. The research and development of these algorithms have made it possible for SLAM to be implemented in many products.

At the same time, with the development of deep learning, a large number of SLAM methods utilizing deep learning technology have been proposed. Some of these methods use deep learning to enhance the performance of some sub-module in classic SLAM algorithms [35–37], while others methods [38–44] adopt innovative end-to-end learning-based frameworks or introduce novel deep networks to implement SLAM. These algorithms have greatly improved the performance in terms of accuracy or stability brought by deep learning. While many algorithms may not achieve real-time performance because of restricted computing resources, as computing power advances in the future, these algorithms establish the groundwork for enhanced SLAM applications in the future.

SLAM localization and 3D reconstruction are generally considered two separate problems. In recent years, there have been some works that combine these two tasks, performing dense 3D reconstruction while spatial localization [45,46], but their main targets are 3D reconstruction but not pose estimation, and even some claimed that they are real-time, but computation is still too heavy to implemented in AR products. In recent two years, thanks to the great works such as NeRF [47] and 3D Gaussian Splatting [48], many works such as iMAP [49], NICE-SLAM [50], Point-SLAM [51], NeRF-SLAM [52], vMAP [53], NeRF-IBVS [54], and GS-SLAM [55], SplaTAM [56], MonoGS [57] use the latest new 3D reconstruction methods with SLAM system to provide real-time high fidelity 3D reconstruction, which is big progress for NeRF and 3D Gaussian Splatting reconstruction to be used in some 3D scanner in real-time.

2.3. Mathematical principle to use SLAM in AR

SLAM can be categorized into lidar SLAM and visual SLAM based on the primary sensors used, as shown in Fig. 2, there are many classical lidar slam algorithms such as [24,32,58], and some works have reviewed lidar SLAM algorithms [59–61] for robots application. However, lidar sensors are impractical for mobile AR devices due to their size and weight, whereas visual cameras are highly suitable. So in this paper, laser SLAM which is used for robots will not be discussed.

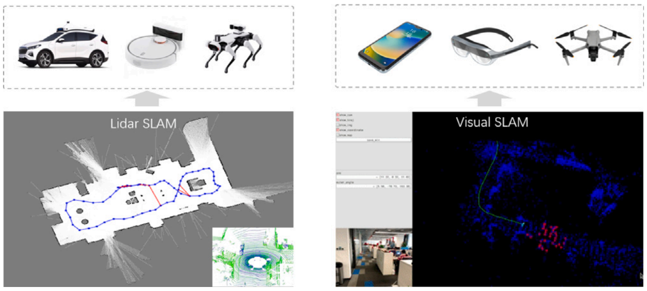

Fig. 2. Lidar SLAM and visual SLAM.

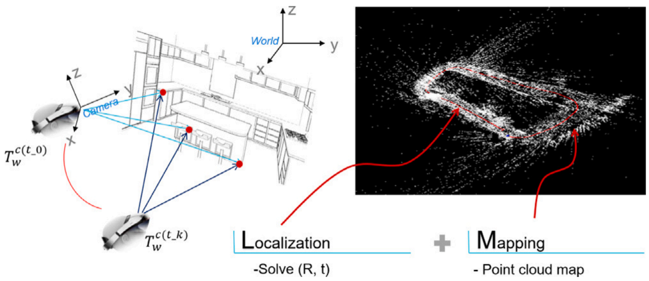

Fig. 3. SLAM for AR.

SLAM algorithm for AR combines the capabilities of SLAM with computer vision and sensor data to accurately track the user's position and orientation in real-time, as shown in Fig. 3. SLAM algorithms for AR typically utilize visual odometry, sparse mapping, and loop detection modules to accurately align virtual objects with the real-world environment, providing users with a realistic and engaging AR experience.

In the context of AR, the main task of SLAM is to consistently provide the real-time position and orientation of the device to the surrounding space. This position and orientation, often referred to as 6 degrees of freedom(6DOF), includes x, y, and z coordinates, as well as the pitch, roll, and yaw angles. The 6DOF is typically represented using a homogeneous matrix, where $\boldsymbol{w}$ represents the world coordinate and $\boldsymbol{c}$ represents the camera coordinate.

$$T_w^c = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{1}$$

There are two methods to determine T_w^c. One approach involves directly calculating the pose relationship between the camera frame and the world coordinate system using a Perspective-n-Point (PnP) method with a known 3D map, or a homography decomposition method with a plane map. Another method, known as visual odometry, involves continuously tracking the relative pose changes between consecutive frames t_0, t_1, t_2, …, t_k to accumulate the pose relationship relative to

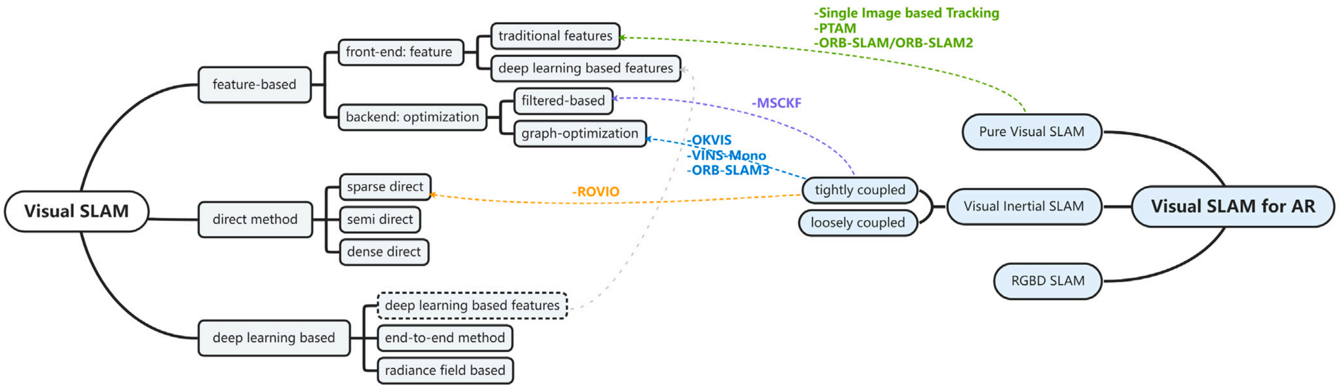

Fig. 4. Visual SLAM and typical algorithm for AR.

the world coordinate system, which can be described as

$$T_w^{c(t_k)} = T_w^{c(t_0)} T_{c(t_0)}^{c(t_1)} T_{c(t_1)}^{c(t_2)} \dots T_{c(t_{k-1})}^{c(t_k)} \tag{2}$$

In AR, the virtual-real overlay effect is achieved by firstly placing virtual objects in the world coordinate system, then binding the virtual camera pose T_W^{vc} based on the real-time 6DOF information provided by SLAM, and finally rendering virtual information by rendering engines (Unreal, Unity) with the virtual camera.

In VST AR, virtual information is observed with the camera's real-time video overlay as background, so the pose obtained from the camera is directly mapped to the virtual camera.

$$T_w^{vc} = T_w^c \tag{3}$$

For the OST AR, virtual information is observed directly through the eyes in the real world rather than through the video overlay method. In the rendering engine's virtual camera, only the virtual information needs to be rendered without overlaying the real physical world's RGB image background. However, the pose used for virtual camera rendering is the eye's pose T_w^{eye} instead of the camera pose T_w^c. Therefore, an external parameter transformation between the eyes and the camera pose T_c^{eye} is required, which is calibrated offline.

$$T_w^{vc} = T_w^{eye} = T_w^c T_c^{eye} \tag{4}$$

To obtain these AR experiences, there are two fundamental prerequisites for utilizing SLAM in AR. The first requirement is the need for high accuracy and stability, while the second requirement is real-time performance and low computational demands due to the limited processing power of mobile AR devices. To achieve high performance in AR, sensor data are also crucial for meeting SLAM requirements. So in this review, SLAM algorithms are deep-dived first and methods of sensor configuration and datasets for AR SLAM are also discussed.

3. SLAM algorithms for AR

The SLAM algorithm used in most AR products relies on visual SLAM. Depending on the device's sensor configuration or application scenario, visual SLAM for AR can be categorized into three types: pure visual SLAM, visual-inertial SLAM, and RGBD-SLAM. The classification depends on whether the algorithm uses inertial measurement unit (IMU) data and depth sensor data.

Pure visual SLAM has the benefit of using only the camera for positioning and tracking. However, there are some drawbacks associated with it. When the motion speed is too fast, it can lead to tracking issues due to factors such as image blur, insufficient frame rate, or weak scene texture. Typically, pure visual SLAM is used in VST AR.

Visual-inertial SLAM leverages the fusion of IMU data, effectively addressing the challenges posed by fast motion, limited frame rate, and weak texture in AR scenes, therefore it is widely used in AR, especially essential for OST AR.

RGBD-SLAM algorithms integrate depth-map to enhance SLAM performance, such as DVO-SLAM [62], ElasticFusion [45], and VINS-RGBD [63]. Depth information provided by depth cameras can offer advantages in absolute scale estimation and precise point cloud map reconstruction for SLAM. However, integrating a depth camera into an AR device can be expensive regarding power consumption, data transmission, and computational resources. As a pioneer, Google explored an RGBD-based solution in the Google Tango AR project, but the outcome was unsatisfactory, leading to the project's closure. Consequently, most AR products, particularly AR glasses, do not incorporate depth cameras. For example, Qualcomm's XR and Google ARCore reference designs do not include a depth camera. Even in cases where a depth camera is present, like in HoloLens and Apple iPhone/iPad, the depth data is solely utilized for 3D reconstruction and not for SLAM pose estimation. Given these factors, this review does not delve into RGBD-based visual SLAM algorithm approaches.

Fig. 4 shows two different classification methods from the perspective of visual SLAM algorithm methodological and sensor configuration for visual SLAM used in AR, as well as typical SLAM algorithms that can currently be applied to AR. In this section, first, an overview of visual SLAM will be provided from a methodological perspective, followed by an introduction to the commonly used pure visual SLAM algorithms for AR, and then a focus will be placed on the visual-inertial SLAM which is mostly used in AR due to its high performance.

3.1. Overview of visual SLAM

From a methodological perspective, visual SLAM algorithms can be classified into three types: feature-based methods, direct methods, and deep learning-based methods. The basic ideas, current status, and existing problems of these three methods will be introduced.

3.1.1. Feature-based method

Feature-based SLAM methods rely on extracting and tracking distinctive features from images to estimate the camera's position and construct a sparse map of the environment. Features can be points, edges, or other salient landmarks that are robustly detected and tracked over time. By associating these features with their corresponding locations in the map, feature-based SLAM algorithms can accurately estimate pose and update the map incrementally. Many famous SLAM algorithms such as PTAM [19],ORB-SLAM [21,22,33],VINS-Mono [31] are feature-based method.

In these methods, on the one hand, many methods contribute to different feature extraction and matching methods. SIFT/SURF [64, 65] stands as the most classical local feature method. Other methods such as ORB [66], BRISK [67], and FREAK [68] are designed to optimize for speed and efficiency by binary based descriptor. SuperPoint [69], and SuperGlue [70] are state-of-the-art feature extraction and matching algorithms by leveraging convolutional neural networks to extract discriminative key points and learn similarity measures to establish correspondences, and some methods such as LoFTR [71] and COTR [72] tried feature matching with transformer networks, but these deep learning based feature extraction and matching methods are not speed efficiency.

On the other hand, in feature-based SLAM methods, pose can be optimized in two ways, which are filter-based optimization and graph optimization. Filter-based SLAM optimization, such as the extended Kalman filter(EKF), uses a recursive Bayesian filter to estimate pose and map in real-time. By updating the belief state with sensor measurements and pose movements, these methods incorporate uncertainty to improve accuracy. Filter-based optimization is computationally efficient, making it suitable for real-time applications. However, it may struggle with non-linearities and large-scale environments. Despite these limitations, filter-based SLAM optimization remains a popular choice due to its simplicity, speed, and ability to provide continuous updates of pose during tracking. Graph-based optimization methods represent the SLAM problem as a graph where nodes correspond to poses and landmarks, and edges represent constraints between them. The optimization problem is then solved by minimizing the error in the graph structure [73]. Graph-based optimization is more flexible and robust than filter-based methods, especially in handling loop closures and large-scale environments. However, it can be computationally more demanding and may require more memory.

Feature-based methods are widely used in AR and also in robotics, and autonomous vehicles, enabling real-time mapping and localization in dynamic environments. But Feature-based methods in visual SLAM require extensive parameter tuning for different camera sensors and user scenarios due to the reliance on specific feature extraction and matching techniques. To achieve optimal results, the parameters of feature-based methods need to be carefully adjusted to suit the characteristics of the camera sensor being used and the specific environment in which the SLAM system operates. This includes parameters related to feature detection, feature matching, outlier rejection, and camera calibration. The parameters tuning process can be time-consuming and requires expertise in understanding the underlying algorithms and their interactions with the camera sensor and environment.

3.1.2. *Direct method*

In the direct method, the camera images are treated as a continuous function of intensity values, and the SLAM system estimates the camera's motion and builds a map by directly optimizing the alignment between consecutive frames without feature extraction. This is achieved by minimizing the photometric error, which measures the difference in intensity values between the current frame and the reference frame. Several famous SLAM algorithms are direct-based. DTAM [20] is a pioneering direct SLAM method that simultaneously estimates the camera trajectory and dense 3D map from a sequence of images. LSD-SLAM [74] is designed for large-scale environments using a monocular camera to track the camera pose directly, and it leverages edge information and photometric consistency to estimate camera motion and reconstruct the environment. SVO [23] is a semi-direct visual odometry method that combines direct and feature-based techniques by leveraging both pixel intensities and sparse features for camera motion estimation. DSO [75] is a direct visual odometry method with a combination of sparse points and a direct method by a non-linear optimization framework.

Direct methods offer advantages in handling textureless or dynamic environments, as they can utilize all available pixel information for localization and mapping. However, they can be computationally intensive and require robust optimization techniques to achieve accurate and real-time results and direct method is more sensitive to lighting changes and image noise compared to feature-based methods because of its basic constant light hypothesis, which assumes that the lighting conditions in the environment remain constant between consecutive frames. Due to these disadvantages, the direct method is rarely used for AR. However, the direct method still has its applications in special scenarios and ongoing developments that may expand its usage in the future.

3.1.3. *Deep learning based method*

SLAM methods that utilize deep learning can be categorized into three approaches. The first approach focuses on enhancing traditional feature-based methods by incorporating deep learning techniques for feature extraction and matching, which is already introduced in the feature-based method, and in this paper, we would like to categorize SLAM algorithms utilizing deep learning in this way as the feature-based method. However, there is potential in utilizing deep learning to extract higher-level semantics instead of feature points for matching, which can pave the way for semantic SLAM. The biggest limitation of this approach is that feature or semantic extraction relies heavily on the inference of large neural networks, making real-time computation difficult to guarantee.

The second approach is end to end deep learning method, which usually trains a pose regression network using neural networks to predict the pose of a given image. For example, in DeepVO [39], camera pose is estimated by a Recurrent Neural Network (RNN) from features learned by a Convolutional Neural Network (CNN), and UnDeepVO [41] uses an unsupervised deep learning scheme with loss function based on spatial and temporal dense information to estimate camera pose. In VLocNet [76], a new CNN architecture for global pose regression and odometry estimation from consecutive images is proposed. [77] propose a novel monocular depth estimation and a pose estimation network with an attention mechanism. The advantage of end to end deep learning-based method that it is simple and does not require manual parameter tuning for each step in the feature-based methods. It only requires training with a large number of images with poses, but the disadvantages are that the accuracy and robustness are still significantly lower than traditional feature-based methods, and it suffers from overfitting to the scene. It also lacks effective theoretical interpretability and is difficult to perform further optimization, and in [78,79], limitations of the deep learning-based camera pose regression method are discussed. Therefore, this type of method has not been widely applied in practice.

The third category to utilize deep learning in SLAM is radiance field such as NeRFs [47] and 3D Gaussian Splatting [48] related methods. iMAP [49] represents the initial effort to utilize implicit neural representations for SLAM. It pushes the boundaries of SLAM and sets a new direction for the field, and many other related algorithms are proposed to leverage neural implicit representations, such as NICE-SLAM [50], Point-SLAM [51],vMAP [53], NeRF-SLAM [52], NeRF-IBVS [54], CP-SLAM [80]. In the last two years, some works have tried to develop SLAM algorithms with explicit volumetric representations based on 3D Gaussian Splatting. SplaTAM [56] first time leverages 3D Gaussians representations to achieve high-fidelity reconstruction from unposed RGBD camera, It implements a straightforward online tracking and mapping system by using a silhouette mask to effectively capture scene density, offering several advantages such as quick rendering and dense optimization. GS-SLAM [55] proposes an adaptive strategy that involves adding or removing noisy 3D Gaussians to reconstruct new scene geometry efficiently, and also an effective coarse-to-fine technique is developed to choose dependable 3D Gaussian representations to optimize camera pose, leading to reduced runtime and robust estimation., MonoGS [57] introduces the initial use of 3D Gaussian Splatting for progressive 3D reconstruction with a single moving monocular or RGB-D camera. It employs Gaussians as the only 3D representation and develops camera tracking for 3DGS through direct optimization against the 3D Gaussians instead of an offline Structure from Motion (SFM), which unifies accurate representation, high-quality rendering, efficient tracking, and mapping together. It achieves 3 fps on a desktop with Intel Core i9 12900 K 3.50 GHz and a NVIDIA GeForce RTX 4090. Although SLAM algorithms based on methods such as NeRF and 3D Gaussian Splatting have provided new possibilities for SLAM research, there are still many limitations to be used in AR products at present, such as poor real-time performance on AR mobile compute platforms, and compared to classical SLAM algorithms for AR, these methods do not offer a higher pose estimation precision, which requires further research and optimization.

Table 1
Pure visual SLAM comparison.

Comparison items	Image based tracking [81,82]	PTAM [19]	ORB-SLAM2 [21]
Accuracy	High (if markers visible)	High (in well-textured environments)	High (in a variety of environments)
Robustness	Low (to occlusions)	Medium	High
Scalability	Limited	Medium	High
Computational cost	Low	Medium	High
Mapping	Marker-based map	Keyframe-based map	Keyframe-based map with loop closure

3.2. Pure visual SLAM for AR

Pure visual SLAM only relies on the camera for positioning and tracking. Almost all visual SLAM algorithms used in AR devices or products are feature-based with the same principle. The key mathematical concept in visual SLAM involves reprojection error, which is the discrepancy between the observed feature points in the images and their reprojected positions based on the estimated 3D points and camera poses. The loss function can be defined as Eq. (5).

$$\mathcal{E}_{\text{visual}} = \sum_{i=1}^{N} \sum_{j=1}^{M} \|\mathbf{z}_{ij} - \pi(\mathbf{T}_i \mathbf{P}_j)\|^2_{\mathbf{\Sigma}_{ij}} \tag{5}$$

- N: Number of camera frames
- M: Number of observed feature points
- $\mathbf{z}_{ij}$: Observed image coordinates of the jth feature in the ith frame
- $\pi(\cdot)$: Camera projection function
- $\mathbf{T}_i$: Transformation matrix (pose) of the ith frame
- $\mathbf{P}_j$: 3D position of the jth feature point
- $\mathbf{\Sigma}_{ij}$: Covariance matrix of feature observation

The primary goal is to estimate the camera poses and the 3D positions of the feature points such that the reprojection error is minimized. To get high performance in AR applications, different optimization methods are used for the items in the loss function. There are three typical algorithms used in AR products, which are single image-based tracking, PTAM (Parallel Tracking and Mapping) [19], and ORB-SLAM/ORB-SLAM2 [21,22]. Table 1 lists a basic comparison of these algorithms. Each algorithm is analyzed as follows.

3.2.1. Single image based tracking

Single image-based tracking is the simplest SLAM algorithm used in AR, the image can be a marker or a printed picture [81,82]. It utilizes a single image to estimate the camera's pose (position and orientation) by decomposing the homography matrix, which represents the projective transformation between two images taken from different viewpoints [83]. From the perspective of optimization for the loss function defined in Eq. (5), the key features of this method are that $\mathbf{P}_j$ are fixed without triangle optimization, only one frame is used so that N=1, and the camera projection function $\pi(\cdot)$ is homography projection which is a special case of projective transformation. Single image-based tracking is the simplest SLAM algorithm used for AR.

To apply this approach, visual features are detected and matched between consecutive frames. These features serve as correspondences that allow for the estimation of the homography matrix. This method makes it possible to estimate the camera's pose without relying on multiple images or additional sensors and is particularly useful in scenarios where only a single image is available or when real-time tracking is required. However, it assumes a planar scene or limited camera motion, as the homography matrix represents a planar projective transformation. Additionally, this approach may be sensitive to noise, outliers, and occlusions, which can affect the accuracy of the pose estimation. Despite these limitations, single image-based SLAM tracking offers a computationally efficient and accessible method for estimating the camera's pose in SLAM systems, making it a valuable tool in many simple AR applications.

3.2.2. PTAM

PTAM is a sophisticated SLAM algorithm that is implemented by optimization for reprojection error defined in Eq. (5). The key contribution is that it operates a parallel processing framework to estimate $\mathbf{T}_i$ and construct 3D points $\mathbf{P}_j$, enabling simultaneous tracking and mapping tasks. This parallel approach enhances the algorithm's efficiency and robustness, particularly in challenging environments.

In the tracking phase, PTAM utilizes feature-based methods to detect and track visual features within the camera's field of view. These features serve as reference points for estimating the camera's pose and motion. By continuously tracking these features across frames, PTAM accurately determines the camera's position and orientation in real-time. Simultaneously, in the mapping phase, PTAM incrementally constructs a sparse 3D map of the environment by estimating the depth of features using the camera's motion and known parameters. This 3D map provides a structural representation of the scene, facilitating accurate localization and navigation. A key strength of PTAM lies in its ability to handle dynamic scenes and camera motion through a keyframe-based approach. Selected frames serve as keyframes not only for mapping but also for tracking and optimization, ensuring robust performance in the face of environmental changes and camera viewpoint variations. Furthermore, PTAM incorporates bundle adjustment techniques to refine camera poses and optimize the 3D map, enhancing accuracy and consistency while reducing drift. However, challenges may arise with large-scale mapping, as increased computational and memory requirements could impact real-time operation and performance in resource-constrained environments. Original PTAM is primarily designed for monocular cameras, which limits its ability to provide an absolute scale estimation, while S-PTAM [26] solved the absolute scale problems and optimized point initialization, mapping, and tracking by stereo constraints.

3.2.3. ORB-SLAM/ORB-SLAM2

ORB-SLAM is the most classical SLAM algorithm that combines feature-based tracking, mapping, and loop closure detection. ORB-SLAM has gained significant popularity in the field of computer vision and robotics due to its robustness, efficiency, and ability to operate in real-time. It has a similar framework as PTAM to estimate pose and 3D points map in parallel threads but introduces more engineering optimization works, especially a robust detection and matching method for observation feature points $\mathbf{z}_{ij}$ in Eq. (5) to increase algorithm robustness.

It utilizes the Oriented FAST and Rotated BRIEF (ORB) feature detector and descriptor to track and match visual features between consecutive frames. The ORB features are designed to be robust to changes in scale, rotation, and viewpoint, making them suitable for SLAM applications. By tracking these features, ORB-SLAM estimates the camera's pose and motion in real-time. In addition to tracking, ORB-SLAM simultaneously builds a 3D map by employing a keyframe-based approach, where selected frames are chosen as keyframes for mapping. These keyframes are used to triangulate the 3D positions of the tracked features, creating a sparse map representation. The map is incrementally updated as new keyframes and features are added. One of the key strengths of ORB-SLAM is its ability to detect and close loops in the map. Loop closure detection is crucial for correcting

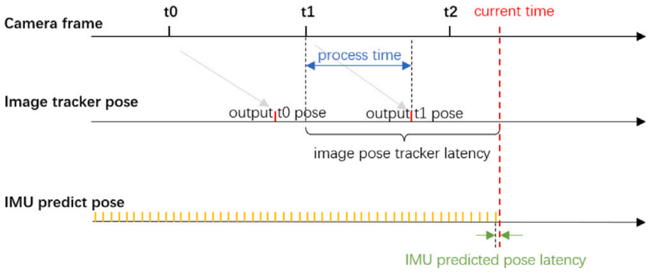

Fig. 5. Output high-frequency SLAM pose by IMU.

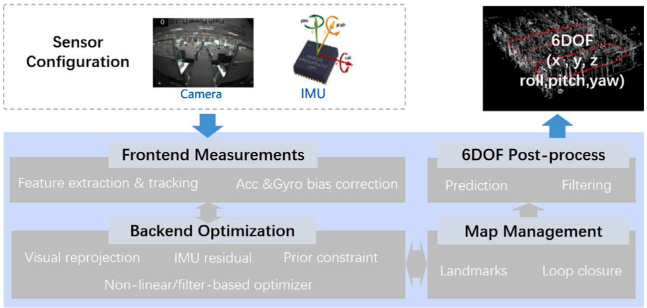

Fig. 6. Typical visual-inertial SLAM algorithm architecture.

accumulated errors and improving the accuracy of the SLAM system. ORB-SLAM achieves loop closure by comparing the current frame with previous keyframes, searching for similarities, and relocalizing the camera within the map. This loop closure process helps to reduce drift and maintain a consistent and accurate map. ORB-SLAM also incorporates bundle adjustment techniques to optimize the camera poses and the 3D map. Furthermore, ORB-SLAM supports both monocular and stereo camera setups, allowing for flexibility in hardware configurations. ORB-SLAM can help to build a full-function SLAM system and a good reference algorithm for many AR products.

3.3. Visual-inertial SLAM for AR

Despite the accomplishments of visual SLAM methods like ORB-SLAM and PTAM, numerous challenges persist in various AR scenarios, mainly including fast motion, limited frame rate, and weak texture environment. Visual-inertial SLAM is a powerful technique that combines visual information from cameras with inertial measurements from an IMU to overcome these challenges.

Fast motion can pose difficulties for visual-based tracking methods, as rapid camera movements can lead to motion blur or loss of visual features. Visual-inertial SLAM can accurately estimate the camera's motion even during fast movements by incorporating IMU data, which provides measurements of acceleration and angular velocity. One of the challenges in AR is the limited frame rate, which can be caused either by a limited camera frame rate or limited computing resources. However, with the help of visual-inertial SLAM, this limitation can be compensated for by utilizing high-frequency measurements (ranging from 100 to 1000HZ) from the IMU. The IMU data provides continuous pose updates based on the camera's motion, resulting in smoother and more accurate tracking, even with lower camera frame rates. As shown in Fig. 5, By using the high-frequency pose output of the IMU, it is possible to achieve a smooth and immersive OST augmented reality experience with very low MTP(Motion-to-Photon) latency. Weak texture in the environment can make it challenging for visual-based methods to detect and track features. However, by combining visual data with inertial measurements, visual-inertial SLAM can better track movement in areas with limited texture or featureless regions. Visual-inertial SLAM enables accurate tracking, even during fast movements, compensates for lower frame rates, and improves tracking performance in areas with limited texture. This makes it a valuable technique for enhancing the quality and realism of AR experiences.

A typical visual-inertial SLAM algorithm architecture is shown in Fig. 6. Initially, sensor data, which includes camera and IMU frames, is fed into the front-end measurement module. This module extracts meaningful visual features and preprocesses the IMU(acceleration, gyro) data. The extracted features and IMU data are then transmitted to the backend optimization module, where target functions with visual re-projection error, IMU residual error, and prior constraint error are optimized to estimate pose and internal variables, such as 3D sparse points and IMU bias. The estimated pose and internal variables can be further used by the map management module. Finally, the 6DOF post-process module utilizes pose prediction and filtering techniques to output a 6DOF pose.

Backend optimization in visual-inertial SLAM typically involves minimizing a loss function that incorporates multiple error terms: visual error, IMU error, and prior error. The visual error $\mathcal{E}_{\text{visual}}$ in Eq. (5) measures how well the estimated 3D points project back onto the image plane compared to the observed feature points.

The IMU error measures the discrepancy between the predicted pose changes from the IMU data and the actual pose changes, and it is often expressed in terms of pre-integrated IMU measurements, which capture the change in rotation, velocity, and position over a time interval between two consecutive states. A typical IMU error loss function can be described as Eq. (6).

$$\mathcal{E}_{\text{imu}} = \sum_{i=1}^{N-1} \|\mathbf{X}_{i+1} - f(\mathbf{X}_i, \mathbf{Z}_i)\|^2_{\mathbf{\Sigma}_i}. \tag{6}$$

- $\mathbf{X}_i = (\mathbf{T}_i, \mathbf{velocity}_i, \mathbf{bias}_i)$: State variable at frame i
- $f(\cdot)$: IMU propagation function
- $\mathbf{Z}_i$: IMU data set between frame of i and $i+1$
- $\mathbf{\Sigma}_i$: Covariance matrix of propagation function

The prior error shown in Eq. (7) incorporates any additional information about the initial states or other known constraints that can help in the estimation process. This error term is useful when we have prior knowledge about the state variables or constraints from historical state variables.

$$\mathcal{E}_{\text{prior}} = \sum_{k \in \text{prior constraints}} \|\mathbf{x}_k - \hat{\mathbf{x}}_k\|^2_{\mathbf{\Sigma}_k} \tag{7}$$

- $\mathbf{x}_k$: State variable constrained by a prior
- $\hat{\mathbf{x}}_k$: Prior estimate of the state variable
- $\mathbf{\Sigma}_k$: Covariance matrix of the prior constraints

The overall optimization problem combines these error terms:

$$\mathcal{E}_{\text{total}} = \mathcal{E}_{\text{visual}} + \mathcal{E}_{\text{imu}} + \mathcal{E}_{\text{prior}} \tag{8}$$

The goal is to minimize this cost function with respect to the state variables (camera poses, feature points, velocities, and biases). This optimization is typically performed using non-linear optimization techniques and filter-based methods. Non-linear optimization techniques like bundle adjustment and factor graph optimization provide high accuracy but can be computationally intensive. Filter-based methods like EKF offer computational efficiency suitable for real-time applications but may sacrifice some accuracy.

Visual-inertial SLAM can be categorized into loosely coupled and tightly coupled according to how the visual and inertial data are integrated and optimized to estimate the state. In loosely coupled methods, visual and inertial measurements are processed separately before being combined at a higher level, in which the loss functions $\mathcal{E}_{\text{visual}}$ and $\mathcal{E}_{\text{imu}}$ for visual and inertial data are typically optimized separately,

Table 2
Visual-Inertial SLAM comparison.

	Comparison items	Method	Frontend	Backend	Key features
	MSCKF [28]	Feature-based	FAST + Optical flow	Visual error + IMU error, EKF	Multi-state estimation; Real-time performance
	ROVIO [30]	Direct method	FAST + Patch feature	Visual error (Photometric) + IMU error, EKF	Robot-centric approach; Real-time direct method
not good for monocular	OKVIS [29]	Feature-based	Harris corner + BRISK	Visual error + IMU error + Prior error, Non-linear optimization	Keyframe-based approach; Sliding window based marginalization
Good	VINS-Mono [31]	Feature-based	Shi-Tomasi corners + Optical flow	Visual error + IMU error + Prior error, Non-linear optimization	Loosely-coupled initialization; unique marginalization strategy; 4DOF global optimization
best	ORB-SLAM3 [33]	Feature-based	Oriented FAST + Rotated BRIEF	Visual error + IMU error + Prior error, Non-linear optimization	Multi-map support; Monocular, stereo, and RGB-D camera support; Loop closure support

and the final state estimate is obtained by fusing the results of these separately minimized loss functions. Loosely coupled systems are more flexible and modular, allowing for easier integration of different sensors and algorithms. However, they may suffer from error accumulation and drift over time due to the lack of tight integration between the visual and inertial measurements, therefore loosely coupled visual-inertial SLAM is rarely used in AR. In tightly coupled methods, visual and inertial measurements are integrated into a single optimization framework to optimize $\mathcal{E}_{\text{total}}$, resulting in a joint estimation process that jointly optimizes all state variables (camera poses, velocities, biases, and 3D points) simultaneously. The tightly coupled methods leverage the complementary nature of visual and inertial data, leading to more accurate and robust state estimates. However, they may be more complex to implement and require careful calibration and synchronization of sensors.

There are several classical algorithms available for AR products, including MSCKF [28], ROVIO [30], OKVIS [29], VINS-Mono/VINS-Fusion [31], and ORB-SLAM3 [33]. Almost all OST AR devices are implemented with engineering optimization based on these algorithms. They have different frontends to extract features and use different ways to construct and optimize loss functions in Eq. (8). Table 2 gives a brief comparison of these visual-inertial algorithms. The detailed features of these algorithms are summarized as follows.

3.3.1. Visual-inertial initialization

Before introducing classical visual-inertial algorithms for AR, visual-inertial initialization is discussed because it is the first procedure for the visual-inertial SLAM algorithm and plays a crucial role in the AR experience. visual-inertial initialization combines visual information from cameras with inertial data from IMU to estimate key variables including scale, gravity direction, camera pose, and IMU bias, and sometimes also including temporal and extrinsic parameters between IMU and camera. Successful visual-inertial initialization is the basis of subsequent SLAM tracking. Slow initialization of visual-inertial in augmented reality applications can lead to long waiting times and significantly reduce user experience. Poor and unstable initialization of visual and IMU states can decrease the accuracy and stability of pose tracking, especially causing drift in the early stages of SLAM tracking.

The visual-inertial initialization method can be roughly divided into three categories: loosely coupled, tightly coupled, and hybrid coupled methods. Loosely coupled methods [84,85] first process visual information separately, then solve different initialization parameters in multiple steps in sequence, and by solving step-by-step and decoupling parameters, the overall computational cost is reduced to achieve fast initialization, but the step-by-step solving process weakens the correlation between different initialization parameters, resulting in a loss of initialization accuracy. Tightly coupled methods such as [86,87] do not process visual information separately but solve all initialization parameters in one step, which can ensure high initialization accuracy. However, due to solving all initialization parameters in one step, the high computational cost and optimization convergence difficulties can lead to a slow overall initialization process. Hybrid coupled methods like [88,89] combine loosely coupled and tightly coupled methods by separately processing visual information, and then solving other initialization parameters in one step, these combined methods not only improve the initialization speed but also ensure the accuracy of the initialization.

In these methods, estimating the feature point cloud typically requires translational motion for triangulation calculations, while estimating the bias of the IMU's accelerometer and gyroscope rotation requires translational and rotational motion. In most cases, a good initialization is easier to obtain when there is significant motion. However, in AR applications, requiring users to have significant motion, particularly translation, when starting the AR application can impact user experience. Therefore, finding a fast initialization under slight motion is challenging but essential in visual-inertial initialization for AR. To address this issue and get fast and stable initialization in AR products, some methods are used to reduce the difficulty of visual-inertial initialization, including off-line high accuracy calibration, hardware time synchronization, stereo or depth-based scale estimation, and low noise IMU sensor implementation. For example, Google ARCore and Qualcomm XR reference design guides both indicate the need for offline calibration and have basic requirements for IMU performance. Most AR glasses use multi-camera solutions to ensure fast initialization although it may result in higher costs.

3.3.2. MSCKF

MSCKF stands out as a pioneer of filter-based optimization on visual and IMU errors. It leverages an Extended Kalman Filter (EKF) to integrate visual and inertial measurements for accurate pose estimation by a multi-state approach. This multi-state approach allows the algorithm to leverage the geometric constraints provided by observing static features from multiple viewpoints over time. Traditional EKF-based approaches include feature points in the state vector for joint estimation with IMU states. When the environment is large, the number of feature points becomes very high, leading to a significant increase in the dimensionality of the state vector. MSCKF does not include feature points in the state vector, instead, it incorporates camera poses at different time instances into the state vector. Since feature points are observed by multiple cameras, geometric constraints are formed among multiple states, which are then used to construct observation models for EKF update. Due to the far fewer camera poses compared to feature points, the dimensionality of the MSCKF state vector is significantly reduced, and computational efficiency is enhanced. In terms of performance, MSCKF demonstrates superior adaptability to fast motion and temporary texture loss compared to pure visual SLAM algorithms. Compared to optimization-based visual-inertial SLAM algorithms, MSCKF offers notable speed advantages. It is particularly well-suited for deployment on embedded platforms with constrained computational resources, making it an excellent choice for AR applications.

3.3.3. ROVIO

ROVIO is an algorithm for monocular visual-inertial odometry that utilizes pixel intensity errors of image patches directly [30]. ROVIO integrates the tracking of multilevel patch features with an iterative EKF by incorporating intensity errors during the update step. Unlike traditional methods that use reprojection error in Eq. (5), ROVIO is a direct-based method that utilizes photometric error and optimizes it through the iterative EKF. By using a completely robot-centric representation of the filter state with a numerically minimal distance representation of features [30], ROVIO prevents significant consistency issues to ensure accuracy and robustness. Because ROVIO is a direct-based method, it can estimate pose even in low-texture environments, and challenging scenarios involving rapid movements or outliers, it can effectively maintain state tracking with minimal yaw and position estimate drift [30].

3.3.4. OKVIS

OKVIS is an advanced SLAM algorithm designed to integrate visual and inertial. One of the key features of OKVIS is its utilization of keyframes, which are selected frames that represent significant changes in the environment. This approach reduces computational complexity by focusing on these keyframes rather than processing every single frame, thereby enhancing efficiency and scalability. Another primary contribution of OKVIS is its robust non-linear optimization framework, which combines both bundle adjustment and marginalization techniques to minimize a cost function that incorporates visual error, IMU error, and prior error. This framework refines the pose estimates by jointly optimizing the states of the keyframes and the associated map points. In marginalization, the oldest key frame is discarded and state variables of the frame are split into two parts. The poses are gradually marginalized based on whether they are keyframes in the sliding window, while the velocity and IMU biases are marginalized immediately. The marginalization of old states helps to maintain a manageable computational load while preserving essential information from previous frames, thus ensuring that the system remains efficient over time. OKVIS is distinguished by its non-linear optimization method and keyframe-oriented feature tracking technique. It is recognized for its real-time capabilities and open-source design, establishing it as a solid base for AR SLAM developers and researchers.

3.3.5. VINS-Mono/VINS-fusion

VINS-Mono is a classical work in the fusion of visual and IMU data. It adopts a sliding window based non-linear optimization framework similar to OKVIS, but VINS-Mono makes some improvements based on this framework. Firstly, it proposes a loosely coupled sensor fusion initialization method to estimate the scale, gravity, velocity, and IMU biases, which is critical for monocular SLAM method. Secondly, VINS-Mono also utilizes marginalization to maintain efficiency but implements a different strategy from OKVIS. The decision to keep the current frame in the window depends on whether the previous frame is a keyframe. If the previous frame is not a keyframe, the current frame is discarded. If it is a keyframe, the oldest frame is marginalized and keeps the current frame in the window. Additionally, the algorithm implements a 4DOF pose graph optimization and loop closure detection to enhance the overall system's global accuracy. VINS-Mono is designed to operate in real-time, providing pose estimates at high frequencies. This makes it very suitable for AR.

As an extension of VINS-Mono, VINS-Fusion [90,91] offers additional features like supporting multiple cameras and performing real-time spatial and temporal calibration. Having stereo or multi-camera support is beneficial for visual-inertial initialization and allows for a larger perspective to detect feature points in the surroundings. One of advantages of VINS-Fusion lies in its ability to calibrate the temporal offset of cameras and IMU timestamps. It estimates the temporal offset by optimizing the time offset, camera position, IMU states, and feature point locations simultaneously. While most AR devices rely on hardware synchronization or offline tools like Kalibr for temporal calibration, online spatial and temporal calibration in uncalibrated devices such as any mobile phone can significantly expand the range of applications for certain AR SLAM algorithms and SDKs.

3.3.6. ORB-SLAM3

ORB-SLAM3 is the enhancement of the ORB-SLAM2 system, and the key feature is supporting visual-inertial with monocular, stereo, and RGB-D cameras. ORB-SLAM3 uses a tightly coupled optimization approach to fuse visual and inertial information. In addition, ORB-SLAM3 has added the function of building multiple maps and defined a more abstract camera model to support fisheye cameras, wide-angle cameras, etc. In the IMU initialization phase, the Maximum-a-Posteriori (MAP) estimation is introduced to achieve fast initialization. In order to support the multi-map function, a new place recognition method with an improved recall rate has been implemented. When tracking is lost, a new sub-map will be built, and it will be merged with the previous sub-map in the loop closure phase. ORB-SLAM3 can continue to run in poor visual information for a long time. Compared with SLAM algorithms that only use the last few seconds of information, ORB-SLAM3 is the first SLAM algorithm that can reuse all previous information in all algorithm stages, achieving higher accuracy.

4. Sensors, datasets, and performance of AR SLAM

AR SLAM algorithm can efficiently run not only depending on the algorithm itself, but also considering the configuration of the sensors, including the type of sensors, specifications, calibration, and other issues. At the same time, datasets play a crucial role in the assessment and enhancement of SLAM algorithms for AR by researchers and developers. Therefore, choosing and constructing SLAM datasets suitable for AR under sensor configuration conditions is key to researching and improving AR SLAM performance.

There are various well-known visual SLAM datasets available, such as KITTI [92], EuRoC [93], TUM [94,95]. These datasets effectively capture the complexities of real-world scenarios that SLAM systems encounter, such as changing lighting conditions, dynamic objects, and diverse environments. Creating visual SLAM datasets specifically for AR involves addressing specific challenges, such as obtaining accurate 6DOF ground-truth data, ensuring synchronization and timestamp accuracy, precise sensor calibration, and designing diverse environments. While most visual SLAM datasets are designed for general use, some are specifically captured in scenarios involving car driving or drones. When evaluating SLAM algorithms for AR, the indoor portion of the EuRoC dataset is commonly used as it is close to the use case of AR devices compared to other datasets.

This section begins by discussing sensor configuration for AR SLAM, followed by introducing typical visual SLAM datasets and then analyzing the methods and challenges involved in creating visual SLAM datasets for AR. Finally, the performance of typical AR SLAM algorithms is evaluated using the indoor data from the EuRoC dataset.

4.1. Sensors configuration for AR SLAM

As mentioned in previous sections, the SLAM algorithm used for AR mainly relies on sensors such as cameras and IMUs. There are two types of sensor configurations used for constructing AR SLAM: visual-inertial configuration with camera and IMU, and pure visual configuration with only camera.

In the creation of datasets for AR scenes, global shutter cameras are typically preferred over rolling shutter cameras for reducing the issue of varying pixel line sampling times in images. Even with rolling shutter RGB cameras commonly used in mobile VST AR applications, the exposure time of captured image frames must be managed, usually kept under 5 ms. Maintaining the camera's data frame rate is crucial to mimic real-time tracking processes, with a minimum of 30

FPS recommended and 60 FPS being even better. The accuracy of a camera's intrinsic parameters significantly impacts the performance of SLAM during camera calibration. Therefore, precise calibration of the camera's intrinsic properties is essential for the sensor utilized in dataset creation. Achieving high-precision calibration involves placing the calibration board at various distances (0.5 m to 5 m) and ensuring that the reprojection error at different image positions is less than 0.5 to 1 pixel.

Regarding IMUs utilized in AR, particularly in OST AR, a high frame rate is essential, typically a minimum of 500 HZ, to guarantee the generation of low-latency 6DOF pose information, which is crucial for maintaining the real-time functionality of the 6DOF employed in AR display engine.

Due to the high requirements for SLAM pose accuracy and smoothness in AR, the setup of the camera and IMU to produce the relevant dataset must also be carefully planned. It is crucial to position the camera and IMU away from heat-emitting sources to minimize alterations in intrinsic and extrinsic parameters resulting from thermal distortion of the equipment. Additionally, it is essential to synchronize all sensors through hardware to guarantee that the readings from the cameras and IMU accurately represent the movement at the same instance.

These fundamental techniques for sensor setup must be taken into account when developing a quality AR SLAM and datasets.

4.2. Visual SLAM datasets

A Visual SLAM dataset typically includes various types of information. It consists of sensor data such as images or video sequences captured by cameras, depth information from depth sensors, and sometimes lidar point clouds. Additionally, the dataset may include IMU and GPS sensors for accurate localization. Ground truth data, such as the camera poses and 3D point clouds of the environment, are also provided. Some datasets may even include semantic labels for objects in the scene. These diverse data types enable researchers to develop and evaluate visual SLAM algorithms effectively. There are some popular visual SLAM algorithm datasets which usually used for visual SLAM algorithm research.

One widely used dataset is the KITTI [92] dataset, which was collected using a car-mounted sensor setup. It includes various sensor data such as stereo images, LIDAR point clouds, and GPS/IMU measurements. The dataset provides ground truth data for camera poses, 3D point clouds, and semantic labels. It is commonly used for evaluating Visual SLAM algorithms in outdoor urban environments. Another notable dataset is the TUM RGB-D dataset [95], which consists of synchronized RGB-D (color and depth) image sequences. It covers different indoor and outdoor scenes, including offices, corridors, and streets. The dataset provides ground truth camera poses and 3D point clouds. It is widely used for evaluating Visual SLAM algorithms that utilize depth information. The EuRoC [93] dataset is another popular choice for Visual SLAM algorithm evaluation. It includes data collected from a micro aerial vehicle (MAV) equipped with various sensors, including cameras, IMU, and laser rangefinder. The dataset covers both indoor and outdoor environments, including office rooms, warehouses, and forests. It provides ground truth data for camera poses and 3D point clouds. The TUM MonoVO [95] dataset focuses on monocular Visual Odometry (VO), which estimates the camera's motion using a single camera. It includes sequences captured from a handheld camera in various indoor and outdoor scenes. The dataset provides ground truth camera poses and is widely used for evaluating monocular Visual SLAM algorithms. The RobotCar [96] dataset is a large-scale dataset collected using a car-mounted sensor setup. It covers urban environments and includes various sensor data such as stereo images, lidar point clouds, and GPS/IMU measurements. The dataset provides ground truth data for camera poses, 3D point clouds, and semantic labels. It is commonly used for evaluating visual SLAM algorithms in challenging outdoor scenarios.

Besides these datasets, there are various others accessible, each with its own characteristics and applications. Some datasets concentrate on particular sensor configurations, like monocular cameras or lidar sensors, whereas others offer multi-modal sensor information. Furthermore, some datasets emphasize specific environments, such as urban or indoor landscapes, while others cover a wide range of scenarios.

4.3. Challenges to build dataset for AR

When developing visual SLAM datasets, three primary factors are typically considered: sensor setup, ground-truth creation, and diverse settings. However, when it comes to AR scenarios, additional challenges and methods must be considered. Unlike driving a car or operating a drone, achieving precise and smooth pose estimation in an AR SLAM dataset is of utmost importance. For instance, a ground-truth error or vibration of 1 cm may not be significant in a car driving scenario, but in AR, it can greatly impact the visual experience negatively. To create a high-quality dataset for AR SLAM, stricter requirements are necessary in terms of ground-truth generation, and varied environments.

To generate a dataset with extremely accurate ground truth, the utilization of GPS/IMU fusion and lidar scanning methods is not suitable. The GPS/IMU fusion method combines GPS and IMU measurements to obtain ground truth camera poses. While GPS provides global position information and IMU captures the camera's motion and orientation, the resulting camera poses have a significant variance due to the larger error in local areas associated with GPS. Therefore, this method is not ideal for generating ground-truth data for AR SLAM. On the other hand, the lidar scanning method utilizes point clouds captured by lidar sensors coupled with cameras to estimate ground-truth pose. However, the pose estimation relies on point cloud alignment using methods like ICP, which are dependent on point cloud precision and optimization techniques. As a result, the estimated ground truth obtained through this method is not truly accurate. For AR SLAM datasets, motion capture systems are employed to generate ground truth data. These systems involve placing markers on the camera or the object being tracked and using multiple cameras to precisely track their positions in 3D space. This approach allows for highly accurate ground truth camera poses. However, the installation of motion capture systems is limited to specific rooms, making it challenging to create large-scale datasets that encompass a wide range of environments.

Since AR can be applied in various settings, it differs from cars or robots that typically operate in specific environments. Therefore, it is crucial to consider environmental variations when evaluating the performance of AR. Several factors need to be taken into account, including different lighting conditions, textures, dynamic environments, and scale. Regarding lighting conditions, this encompasses scenes with varying levels of illumination, such as bright sunlight, dimly lit rooms, or artificial lighting. This allows for the assessment of the SLAM algorithm's robustness in handling different lighting situations. In terms of textures, environments with rich, poor, smooth, or reflective textures enable the evaluation of the SLAM algorithm's ability to handle diverse surface characteristics and adapt to various environments. The real world is typically a dynamic environment with moving objects or people, so datasets with dynamic objects help assess the SLAM algorithm's accuracy in accurately tracking and mapping the environment while dealing with dynamic elements. Additionally, considering different scales helps evaluate the SLAM algorithm's performance in different spatial contexts, ranging from small indoor environments to large outdoor areas. By incorporating these variations into the dataset, researchers can assess the robustness, accuracy, and adaptability of the visual SLAM algorithm in different AR/VR scenarios.

4.4. Performance evaluation

The indoor portion of the EuRoC dataset is a subset of the larger EuRoC dataset that focuses on indoor environments. The indoor scenes

Table 3
Benchmark of classical SLAM algorithm for AR on EuRoC indoor.

Algorithm	RMSE translation (m)	RMSE rotation (deg/m)
PTAM	0.45	3.3
ORB-SLAM	0.08	3.1
MSCKF	0.41	1.9
ROVIO	0.32	1.7
OKVIS	0.23	1.6
VINS-Mono	0.11	0.9
ORB-SLAM3	0.03	0.4

in the EuRoC dataset include different types of environments such as office rooms, corridors, and other indoor spaces. The dataset provides synchronized data from multiple sensors, including stereo images, IMU measurements, and laser rangefinder scans. The stereo images capture the visual information of the environment, while the IMU measurements provide information about motion and orientation. It is very close to classical AR SLAM sensor configuration and environment setting. Therefore, we analyzed the benchmark of classical SLAM algorithms which are discussed in the previous section on this dataset with same mono camera and inertial data configuration. Since the evaluation is performed for AR SLAM, both translation and rotation are evaluated.

Table 3 shows that visual-inertial SLAM Algorithm (MSCKF, ROVIO, OKVIS, VINS-Mono, ORB-SLAM3) generally outperform pure visual SLAM algorithms (PTAM, ORB-SLAM) in terms of accuracy, and optimization based VIO algorithms (OKVIS, VINS-Mono, ORB-SLAM3) achieve better accuracy than filtered based ones(MSCKF, ROVIO) due to joint optimization of visual and inertial measurements while requiring more computing resources. Among tightly coupled VIO algorithms, ORB-SLAM3 demonstrates superior accuracy, especially in rotation estimation. In fact, this is just a macro-level comparison of these algorithms. In practical AR products, the performance of the algorithms is influenced by hardware and engineering factors, and a comprehensive consideration of multiple factors is usually made for selection.

5. Summary

In summary, AR has emerged as a technology that enhances user perception and interaction with the real world by overlaying virtual objects, in which SLAM algorithms play a crucial role in enabling AR applications by allowing devices to understand their position and orientation. This paper has provided a comprehensive overview and exploration of SLAM algorithms for AR. The paper discussed various types of SLAM algorithms, with a particular focus on visual SLAM algorithms, including feature-based methods, direct methods, and deep learning-based methods. Each approach was examined in terms of its advantages and limitations, providing readers with a comprehensive understanding of the different techniques available. Then, the paper delved into typical visual SLAM and visual-inertial SLAM algorithms for AR. Furthermore, the sensor configuration for AR SLAM is discussed. Finally, datasets and the performance of typical SLAM algorithms for AR are evaluated and discussed. Overall, this review serves as a valuable resource for researchers and practitioners interested in understanding the advancements and challenges in SLAM algorithms for AR. It provides insights into the current state of the field and offers directions for future research and development. By addressing the limitations and challenges, researchers can work towards improving SLAM algorithms for AR, ultimately enhancing the user experience and expanding the potential applications of AR technology.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This paper is supported by Lenovo-SJTU AI Joint Lab. The authors would like to thank all researchers from the Lenovo Research AR project for their discussion and support.

References

[1] S. Mystakidis, A. Christopoulos, N. Pellas, A systematic mapping review of augmented reality applications to support STEM learning in higher education, Educ. Inf. Technol. 27 (2) (2022) 1883–1927.
[2] C.-Y. Lin, H.-C. Chai, J.-y. Wang, C.-J. Chen, Y.-H. Liu, C.-W. Chen, C.-W. Lin, Y.-M. Huang, Augmented reality in educational activities for children with disabilities, Displays 42 (2016) 51–54.
[3] A.W.K. Yeung, A. Tosevska, E. Klager, F. Eibensteiner, D. Laxar, J. Stoyanov, M. Glisic, S. Zeiner, S.T. Kulnik, R. Crutzen, et al., Virtual and augmented reality applications in medicine: analysis of the scientific literature, J. Med. Internet Res. 23 (2) (2021) e25499.
[4] J. Song, J. Wang, S. Zhu, H. Hu, M. Zhai, J. Xie, H. Gao, Mixture reality-based assistive system for visually impaired people, Displays 78 (2023) 102449.
[5] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, J.J. Leonard, Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age, IEEE Trans. Robot. 32 (6) (2016) 1309–1332.
[6] T. Taketomi, H. Uchiyama, S. Ikeda, Visual SLAM algorithms: A survey from 2010 to 2016, IPSJ Trans. Comput. Vis. Appl. 9 (1) (2017) 1–11.
[7] B. Alsadik, S. Karam, The simultaneous localization and mapping (SLAM)-an overview, Surv. Geospat. Eng. J. 2 (2021) 34–45.
[8] A. Macario Barros, M. Michel, Y. Moline, G. Corre, F. Carrel, A comprehensive survey of visual slam algorithms, Robotics 11 (1) (2022) 24.
[9] S. Li, D. Zhang, Y. Xian, B. Li, T. Zhang, C. Zhong, Overview of deep learning application on visual SLAM, Displays 74 (2022) 102298.
[10] T.T.O. Takleh, N.A. Bakar, S.A. Rahman, R. Hamzah, Z. Aziz, A brief survey on SLAM methods in autonomous vehicle, Int. J. Eng. Technol. 7 (4) (2018) 38–43.
[11] S. Zhang, S. Zhao, D. An, J. Liu, H. Wang, Y. Feng, D. Li, R. Zhao, Visual SLAM for underwater vehicles: A survey, Comp. Sci. Rev. 46 (2022) 100510.
[12] M. Billinghurst, A. Clark, G. Lee, et al., A survey of augmented reality, Found. Trends® Hum.–Comput. Interact. 8 (2–3) (2015) 73–272.
[13] E. Marchand, H. Uchiyama, F. Spindler, Pose estimation for augmented reality: a hands-on survey, IEEE Trans. Vis. Comput. Graph. 22 (12) (2015) 2633–2651.
[14] L. Jinyu, Y. Bangbang, C. Danpeng, W. Nan, Z. Guofeng, B. Hujun, Survey and evaluation of monocular visual-inertial SLAM algorithms for augmented reality, Virtual Real. Intell. Hardw. 1 (4) (2019) 386–410.
[15] Wikitude augmented reality, 2024, https://www.wikitude.com/. (Accessed 08 March 2024).
[16] PTC vuforia, 2024, https://www.ptc.com/en/products/vuforia. (Accessed 08 March 2024).
[17] EasyAR, 2024, https://www.easyar.com/. (Accessed 08 March 2024).
[18] A.J. Davison, I.D. Reid, N.D. Molton, O. Stasse, MonoSLAM: Real-time single camera SLAM, IEEE Trans. Pattern Anal. Mach. Intell. 29 (6) (2007) 1052–1067.
[19] G. Klein, D. Murray, Parallel tracking and mapping for small AR workspaces, in: 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, IEEE, 2007, pp. 225–234.
[20] R.A. Newcombe, S.J. Lovegrove, A.J. Davison, DTAM: Dense tracking and mapping in real-time, in: 2011 International Conference on Computer Vision, IEEE, 2011, pp. 2320–2327.
[21] R. Mur-Artal, J.M.M. Montiel, J.D. Tardos, ORB-SLAM: a versatile and accurate monocular SLAM system, IEEE Trans. Robot. 31 (5) (2015) 1147–1163.
[22] R. Mur-Artal, J.D. Tardós, Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras, IEEE Trans. Robot. 33 (5) (2017) 1255–1262.
[23] C. Forster, M. Pizzoli, D. Scaramuzza, SVO: Fast semi-direct monocular visual odometry, in: 2014 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2014, pp. 15–22.
[24] J. Zhang, S. Singh, LOAM: Lidar odometry and mapping in real-time, in: Robotics: Science and Systems, Vol. 2, Berkeley, CA, 2014, pp. 1–9.
[25] A. Pumarola, A. Vakhitov, A. Agudo, A. Sanfeliu, F. Moreno-Noguer, PL-SLAM: Real-time monocular visual SLAM with points and lines, in: 2017 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2017, pp. 4503–4508.
[26] T. Pire, T. Fischer, G. Castro, P. De Cristóforis, J. Civera, J.J. Berlles, S-PTAM: Stereo parallel tracking and mapping, Robot. Auton. Syst. 93 (2017) 27–42.
[27] A.J. Ben Ali, M. Kouroshli, S. Semenova, Z.S. Hashemifar, S.Y. Ko, K. Dantu, Edge-SLAM: Edge-assisted visual simultaneous localization and mapping, ACM Trans. Embed. Comput. Syst. 22 (1) (2022) 1–31.

[28] A.I. Mourikis, S.I. Roumeliotis, A multi-state constraint Kalman filter for vision-aided inertial navigation, in: Proceedings 2007 IEEE International Conference on Robotics and Automation, IEEE, 2007, pp. 3565–3572.

[29] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, P. Furgale, Keyframe-based visual–inertial odometry using nonlinear optimization, Int. J. Robot. Res. 34 (3) (2015) 314–334.

[30] M. Bloesch, S. Omari, M. Hutter, R. Siegwart, Robust visual inertial odometry using a direct EKF-based approach, in: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2015, pp. 298–304.

[31] T. Qin, P. Li, S. Shen, Vins-mono: A robust and versatile monocular visual-inertial state estimator, IEEE Trans. Robot. 34 (4) (2018) 1004–1020.

[32] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, D. Rus, Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2020, pp. 5135–5142.

[33] C. Campos, R. Elvira, J.J.G. Rodríguez, J.M. Montiel, J.D. Tardós, Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam, IEEE Trans. Robot. 37 (6) (2021) 1874–1890.

[34] J. Li, X. Pan, G. Huang, Z. Zhang, N. Wang, H. Bao, G. Zhang, RD-VIO: Robust visual-inertial odometry for mobile augmented reality in dynamic environments, IEEE Trans. Vis. Comput. Graphics (2024).

[35] M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, A.J. Davison, Codeslam—learning a compact, optimisable representation for dense visual slam, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2560–2568.

[36] L. Xiao, J. Wang, X. Qiu, Z. Rong, X. Zou, Dynamic-SLAM: Semantic monocular visual localization and mapping based on deep learning in dynamic environment, Robot. Auton. Syst. 117 (2019) 1–16.

[37] J. Czarnowski, T. Laidlow, R. Clark, A.J. Davison, Deepfactors: Real-time probabilistic dense monocular slam, IEEE Robot. Autom. Lett. 5 (2) (2020) 721–728.

[38] A. Kendall, M. Grimes, R. Cipolla, Posenet: A convolutional network for real-time 6-dof camera relocalization, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 2938–2946.

[39] S. Wang, R. Clark, H. Wen, N. Trigoni, Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks, in: 2017 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2017, pp. 2043–2050.

[40] K. Tateno, F. Tombari, I. Laina, N. Navab, Cnn-slam: Real-time dense monocular slam with learned depth prediction, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 6243–6252.

[41] R. Li, S. Wang, Z. Long, D. Gu, Undeepvo: Monocular visual odometry through unsupervised deep learning, in: 2018 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2018, pp. 7286–7291.

[42] R. Li, S. Wang, D. Gu, DeepSLAM: A robust monocular SLAM system with unsupervised deep learning, IEEE Trans. Ind. Electron. 68 (4) (2020) 3577–3587.

[43] Q. Jia, Y. Pu, J. Chen, J. Cheng, C. Liao, X. Yang, D 2 VO: Monocular deep direct visual odometry, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2020, pp. 10158–10165.

[44] Z. Teed, L. Lipson, J. Deng, Deep patch visual odometry, Adv. Neural Inf. Process. Syst. 36 (2024).

[45] T. Whelan, R.F. Salas-Moreno, B. Glocker, A.J. Davison, S. Leutenegger, Elastic-Fusion: Real-time dense SLAM and light source estimation, Int. J. Robot. Res. 35 (14) (2016) 1697–1716.

[46] M. Labbé, F. Michaud, RTAB-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation, J. Field Robot. 36 (2) (2019) 416–446.

[47] B. Mildenhall, P.P. Srinivasan, M. Tancik, J.T. Barron, R. Ramamoorthi, R. Ng, Nerf: Representing scenes as neural radiance fields for view synthesis, Commun. ACM 65 (1) (2021) 99–106.

[48] B. Kerbl, G. Kopanas, T. Leimkühler, G. Drettakis, 3D gaussian splatting for real-time radiance field rendering, ACM Trans. Graph. 42 (4) (2023) 1–14.

[49] E. Sucar, S. Liu, J. Ortiz, A.J. Davison, iMAP: Implicit mapping and positioning in real-time, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 6229–6238.

[50] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M.R. Oswald, M. Pollefeys, Nice-slam: Neural implicit scalable encoding for slam, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 12786–12796.

[51] E. Sandström, Y. Li, L. Van Gool, M.R. Oswald, Point-slam: Dense neural point cloud-based slam, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 18433–18444.

[52] A. Rosinol, J.J. Leonard, L. Carlone, Nerf-slam: Real-time dense monocular slam with neural radiance fields, in: 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2023, pp. 3437–3444.

[53] X. Kong, S. Liu, M. Taher, A.J. Davison, Vmap: Vectorised object mapping for neural field slam, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 952–961.

[54] Y. Wang, Y. Yan, D. Shi, W. Zhu, J. Xia, T. Jeff, S. Jin, K. Gao, X. Li, X. Yang, NeRF-IBVS: Visual servo based on NeRF for visual localization and navigation, Adv. Neural Inf. Process. Syst. 36 (2024).

[55] C. Yan, D. Qu, D. Xu, B. Zhao, Z. Wang, D. Wang, X. Li, GS-SLAM: Dense visual SLAM with 3D Gaussian splatting, in: CVPR, 2024.

[56] N. Keetha, J. Karhade, K.M. Jatavallabhula, G. Yang, S. Scherer, D. Ramanan, J. Luiten, SplaTAM: Splat, track & map 3D Gaussians for dense RGB-D SLAM, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024.

[57] H. Matsuki, R. Murai, P.H.J. Kelly, A.J. Davison, Gaussian splatting SLAM, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024.

[58] GitHub:Cartographer-project/cartographer, 2017, https://github.com/cartographer-project/cartographer. (Accessed 10 March 2024).

[59] J.M. Santos, D. Portugal, R.P. Rocha, An evaluation of 2D SLAM techniques available in robot operating system, in: 2013 IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR, IEEE, 2013, pp. 1–6.

[60] M.U. Khan, S.A.A. Zaidi, A. Ishtiaq, S.U.R. Bukhari, S. Samer, A. Farman, A comparative survey of lidar-slam and lidar based sensor technologies, in: 2021 Mohammad Ali Jinnah University International Conference on Computing, MAJICC, IEEE, 2021, pp. 1–8.

[61] D. Van Nam, K. Gon-Woo, Solid-state LiDAR based-SLAM: A concise review and application, in: 2021 IEEE International Conference on Big Data and Smart Computing, BigComp, IEEE, 2021, pp. 302–305.

[62] C. Kerl, J. Sturm, D. Cremers, Dense visual SLAM for RGB-D cameras, in: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2013, pp. 2100–2106.

[63] Z. Shan, R. Li, S. Schwertfeger, RGBD-inertial trajectory estimation and mapping for ground robots, Sensors 19 (10) (2019) 2251.

[64] D.G. Lowe, Object recognition from local scale-invariant features, in: Proceedings of the Seventh IEEE International Conference on Computer Vision, Vol. 2, IEEE, 1999, pp. 1150–1157.

[65] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, Speeded-up robust features (SURF), Comput. Vis. Image Underst. 110 (3) (2008) 346–359.

[66] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, ORB: An efficient alternative to SIFT or SURF, in: 2011 International Conference on Computer Vision, IEEE, 2011, pp. 2564–2571.

[67] S. Leutenegger, M. Chli, R.Y. Siegwart, BRISK: Binary robust invariant scalable keypoints, in: 2011 International Conference on Computer Vision, IEEE, 2011, pp. 2548–2555.

[68] A. Alahi, R. Ortiz, P. Vandergheynst, Freak: Fast retina keypoint, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2012, pp. 510–517.

[69] D. DeTone, T. Malisiewicz, A. Rabinovich, Superpoint: Self-supervised interest point detection and description, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2018, pp. 224–236.

[70] P.-E. Sarlin, D. DeTone, T. Malisiewicz, A. Rabinovich, Superglue: Learning feature matching with graph neural networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 4938–4947.

[71] J. Sun, Z. Shen, Y. Wang, H. Bao, X. Zhou, LoFTR: Detector-free local feature matching with transformers, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 8922–8931.

[72] W. Jiang, E. Trulls, J. Hosang, A. Tagliasacchi, K.M. Yi, Cotr: Correspondence transformer for matching across images, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 6207–6217.

[73] G. Grisetti, R. Kümmerle, C. Stachniss, W. Burgard, A tutorial on graph-based SLAM, IEEE Intell. Transp. Syst. Mag. 2 (4) (2010) 31–43.

[74] J. Engel, T. Schöps, D. Cremers, LSD-SLAM: Large-scale direct monocular SLAM, in: European Conference on Computer Vision, Springer, 2014, pp. 834–849.

[75] J. Engel, V. Koltun, D. Cremers, Direct sparse odometry, IEEE Trans. Pattern Anal. Mach. Intell. 40 (3) (2017) 611–625.

[76] A. Valada, N. Radwan, W. Burgard, Deep auxiliary learning for visual localization and odometry, in: 2018 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2018, pp. 6939–6946.

[77] W. Zhao, Y. Wang, Z. Wang, R. Li, P. Xiao, J. Wang, R. Guo, Self-supervised deep monocular visual odometry and depth estimation with observation variation, Displays 80 (2023) 102553.

[78] R. Li, S. Wang, D. Gu, Ongoing evolution of visual SLAM from geometry to deep learning: Challenges and opportunities, Cogn. Comput. 10 (2018) 875–889.

[79] T. Sattler, Q. Zhou, M. Pollefeys, L. Leal-Taixe, Understanding the limitations of cnn-based absolute camera pose regression, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 3302–3312.

[80] J. Hu, M. Mao, H. Bao, G. Zhang, Z. Cui, Cp-slam: Collaborative neural point-based slam system, Adv. Neural Inf. Process. Syst. 36 (2024).

[81] M. Hirzer, Marker detection for augmented reality applications, in: Seminar/Project Image Analysis Graz, Vol. 25, 2008.

[82] A.K. Dash, S.K. Behera, D.P. Dogra, P.P. Roy, Designing of marker-based augmented reality learning environment for kids using convolutional neural network architecture, Displays 55 (2018) 46–54.

[83] S.J. Prince, K. Xu, A.D. Cheok, Augmented reality camera tracking with homographies, IEEE Comput. Graph. Appl. 22 (6) (2002) 39–45.

[84] A. Martinelli, Closed-form solution of visual-inertial structure from motion, Int. J. Comput. Vis. 106 (2) (2014) 138–152.

[85] J. Kaiser, A. Martinelli, F. Fontana, D. Scaramuzza, Simultaneous state initialization and gyroscope bias calibration in visual inertial aided navigation, IEEE Robot. Autom. Lett. 2 (1) (2016) 18–25.

[86] R. Mur-Artal, J.D. Tardós, Visual-inertial monocular SLAM with map reuse, IEEE Robot. Autom. Lett. 2 (2) (2017) 796–803.

[87] T. Qin, S. Shen, Robust initialization of monocular visual-inertial estimation on aerial robots, in: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2017, pp. 4225–4232.

[88] Z. Yang, S. Shen, Monocular visual–inertial state estimation with online initialization and camera–IMU extrinsic calibration, IEEE Trans. Autom. Sci. Eng. 14 (1) (2016) 39–51.

[89] C. Campos, J.M. Montiel, J.D. Tardós, Inertial-only optimization for visual-inertial initialization, in: 2020 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2020, pp. 51–57.

[90] T. Qin, S. Cao, J. Pan, S. Shen, A general optimization-based framework for global pose estimation with multiple sensors, 2019, arXiv preprint arXiv:1901.03642.

[91] T. Qin, S. Shen, Online temporal calibration for monocular visual-inertial systems, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2018, pp. 3662–3669.

[92] A. Geiger, P. Lenz, C. Stiller, R. Urtasun, Vision meets robotics: The kitti dataset, Int. J. Robot. Res. 32 (11) (2013) 1231–1237.

[93] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M.W. Achtelik, R. Siegwart, The EuRoC micro aerial vehicle datasets, Int. J. Robot. Res. 35 (10) (2016) 1157–1163.

[94] D. Schubert, T. Goll, N. Demmel, V. Usenko, J. Stückler, D. Cremers, The TUM VI benchmark for evaluating visual-inertial odometry, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2018, pp. 1680–1687.

[95] Computer vision group - datasets - monocular visual odometry dataset, 2016, https://cvg.cit.tum.de/data/datasets/mono-dataset. (Accessed 18 March 2024).

[96] W. Maddern, G. Pascoe, C. Linegar, P. Newman, 1 year, 1000 km: The oxford robotcar dataset, Int. J. Robot. Res. 36 (1) (2017) 3–15.