Hindawi
Journal of Sensors
Volume 2021, Article ID 2054828, 26 pages
https://doi.org/10.1155/2021/2054828

Research Article

Visual and Visual-Inertial SLAM: State of the Art, Classification, and Experimental Benchmarking

Myriam Servières,[1,2,3] **Valérie Renaudin**,[3,4] **Alexis Dupuis**,[1,3] **and Nicolas Antigny**[1,3,4]

[1]*Centrale Nantes, Nantes 44321, France*
[2]*AAU-CRENAU, ENSA, Nantes, France*
[3]*IRSTV, Nantes 44321, France*
[4]*AME-GEOLOC, IFSTTAR, Univ. Gustave Eiffel, Bouguenais 44344, France*

Correspondence should be addressed to Myriam Servières; myriam.servieres@ec-nantes.fr

Received 11 December 2019; Revised 1 December 2020; Accepted 9 January 2021; Published 25 February 2021

Academic Editor: Stelios M. Potirakis

Simultaneous Localization and Mapping is now widely adopted by many applications, and researchers have produced very dense literature on this topic. With the advent of smart devices, embedding cameras, inertial measurement units, visual SLAM (vSLAM), and visual-inertial SLAM (viSLAM) are enabling novel general public applications. In this context, this paper conducts a review of popular SLAM approaches with a focus on vSLAM/viSLAM, both at fundamental and experimental levels. It starts with a structured overview of existing vSLAM and viSLAM designs and continues with a new classification of a dozen main state-of-the-art methods. A chronological survey of viSLAM's development highlights the historical milestones and presents more recent methods into a classification. Finally, the performance of vSLAM is experimentally assessed for the use case of pedestrian pose estimation with a handheld device in urban environments. The performance of five open-source methods Vins-Mono, ROVIO, ORB-SLAM2, DSO, and LSD-SLAM is compared using the EuRoC MAV dataset and a new visual-inertial dataset corresponding to urban pedestrian navigation. A detailed analysis of the computation results identifies the strengths and weaknesses for each method. Globally, ORB-SLAM2 appears to be the most promising algorithm to address the challenges of urban pedestrian navigation, tested with two datasets.

1. Introduction

The Simultaneous Localization and Mapping (SLAM) problem has been one of the most active research subjects since its formulation in the 1980s [1, 2]. SLAM's goal is to obtain a global and consistent estimate of a device's path while reconstructing a map of the surrounding environment. The coupling between these two tasks, initially considered as the core issue, was soon discovered to be the real strength of SLAM methods. This duality has also encouraged its diversification. By dosing the importance given to mapping or to localization, SLAM has been pushed away from the sole robotics field and became a reference to solve problems of many different natures: from micro aerial vehicles [3] to augmented reality (AR) on a smartphone [4, 5].

Higher expectations were added to the existing SLAM algorithm (real-time, cheap sensors) leading to a new research field on SLAM. Visual SLAM (vSLAM) using solely cameras and visual-inertial SLAM (viSLAM) using inertial measurement units (IMUs) give a good illustration of these new SLAM strategies. vSLAM has probably attracted most of the research over the last decades. Cameras capture numerous data about the observed environment that can be extracted and used for SLAM processing. These cameras are also among the cheapest sensors. Their presence on most of the smart devices today supports the ongoing development of novel applications that target the general public.

Because there exist many different SLAM methods targeting different objectives, comparing them is not easy. Choosing the best-suited method for a specific application

requires a good knowledge of the ins and outs of SLAM as well as a global understanding of state-of-the-art SLAM strategies. The performance of a method depends on the application context and the challenges to be addressed. Globally, SLAM tends to be wrongly considered as an almighty technology but real-life implementation raises many issues in terms of computational limitations, noise mitigation, or even user-friendliness. This is only a selection of the difficulties to overcome.

This paper is aimed at classifying existing vSLAM and viSLAM methods. The proposed transverse classification is based on technical and application-oriented criteria. The review of SLAM methods continues with a historical presentation of vSLAM and viSLAM development. The analysis is completed by running five selected state-of-the-art SLAM methods, which have been chosen to represent the diversity of existing SLAM designs, on two different datasets. These methods best address the use case of pedestrian pose estimation in the urban environment. This experimental benchmark is conducted on a renowned public dataset EuRoC [6] and completed with a new visual-inertial dataset, which has been recorded with smart devices held in hand by a pedestrian in the city center of Nantes in France (IRSTV dataset).

The main motivation supporting this review and benchmark is to assess vSLAM and viSLAM methods in the specific context of pedestrian mobility in the city with augmented reality (AR) used along the journey. Its outcomes should ease the choice of the most suitable methods to estimate the pose of a handheld smart device in this context. Pedestrians' hands are performing 6DoF motions. Looking for popular benchmarking datasets, it was found that these movements are similar to those of micro air vehicles. The paper starts with a classification of the methods derived from the literature according to their characteristics and their robustness to various scenarios (Section 6) to select a dataset. The closest ones to the pedestrian mobility requirements and context were chosen for comparison with the newly introduced IRSTV pedestrian dataset (Section 7).

Section 2 lists existing surveys and benchmarks of SLAM methods with different approaches than the one adopted in this work. Section 3 describes the first level of SLAM algorithms' design: hardware and general software choices. Section 4 describes the general architecture of the vSLAM algorithm. It identifies and details four constitutive "blocks." Section 5 gives an overview of the SLAM history, which is divided into three ages. Section 6 presents a new classification of vSLAM and viSLAM methods. Finally, Section 7 presents the conducted experimental benchmark based on the EuRoC and IRSTV dataset. A detailed analysis of the SLAM results over the selected dataset completes this section.

2. Existing Surveys and Benchmarks

Several survey papers present a snapshot of the state of research on SLAM at a given time. Various papers present other experimental benchmarks. They inform about the performance of the most famous SLAM methods on the given dataset. This section lists some of these papers to conduct a theoretical benchmark. They are also listed to support the comparison of the experimental benchmark, conducted in Section 7, with other state-of-the-art assessments.

2.1. Survey Papers. An interesting and complete, albeit a little old, review on the vSLAM algorithm can be found in [7]. In [8], the authors introduce some of the main differences between state-of-the-art SLAM methods as well as the most famous algorithms in vSLAM, with a very useful introduction to viSLAM. However, as a survey, it mainly provides high-level explanations on the subject. An overview limited to visual odometry and visual SLAM can be found in [9]. Two founding papers to understand the origin of SLAM research are in [10, 11]. They also mainly concentrate on visual odometry with a subpart on viSLAM. The 2006 papers by Durrant-Whyte and Bailey [12, 13] provide rich tutorials on viSLAM. They contain educational and detailed presentations of the mathematical formulation of the SLAM problem but are lacking an updated presentation of recent vSLAM. The paper by Cadena et al. [14] can be considered as a handbook in the field of viSLAM. A recent review [15] lists and classifies filtered-based and optimization-based viSLAM algorithms and compares them using the EuRoc dataset. It is found that none of these papers provides a complete presentation of the subject, i.e., from technical and historical trivia to actual performance comparison. Therefore, the work presented in this paper intends to facilitate the comparison of new research works with SLAM and to assist future research in pose estimation. State of the art presents the main vSLAM methods to explain the impact of design and hardware choices on the performance. The vSLAM/viSLAM classification provides both an overview and a comparison of the diversity of the many existing implementations. Moreover, the classification that is subsequently proposed in Section 6 groups together characteristics that are partially found in other reviews but not necessarily presented all together.

2.2. Benchmarks. A first benchmark [16] focuses on RGB-D SLAM only. Another benchmark [3], dedicated to visual-inertial methods, evaluates tightly coupled Visual Inertial Odometry (VIO) and viSLAM methods on multiple platforms to simulate real-life applications with flying drones. The tests are only performed on the EuRoC dataset (i.e., medium indoor environment). It provides one of the most complete benchmarks on viSLAM algorithms by comparing the accuracy, memory and CPU usage, and computation time of six state-of-the-art algorithms. The S-MSCKF paper [17] compares some viSLAM methods as well. vSLAM methods are often tested on the new college [18], TUM monocular [19], or TUM RGB-D dataset [16] that do not include inertial data. This dataset cannot be used to compare the latest viSLAM methods.

3. Hardware and General Design Choices

SLAM and visual odometry (VO) are often synonyms in the literature because they are both potential choices to solve similar problems, but they target different objectives. VO focuses on estimating the path of a camera in real time. It is done sequentially, each time a new frame is captured. VO

provides only local/relative estimates, and the path is refined online with windowed optimization. On the contrary, SLAM provides a global and consistent estimate of a device's path. The detection of loop closure reduces the drift in both the map and the trajectory estimates by performing bundle adjustment (BA). To simplify, VO and vSLAM act similarly until closing a loop. VO is often used as a building block for vSLAM, which also borrows 3D reconstruction methods from Structure from Motion (SfM) approaches.

Pure VO and vSLAM are both conceivable in applications where building an accurate map is not required.

3.1. Hardware for Visual SLAM. Classically, vSLAM uses three hardware types: monocular cameras, stereo cameras, and RGB-D cameras. viSLAM has drawn increased interest recently because IMU and cameras have complementary features. Cameras are accurate in slow motion and provide a rich source of information, but they suffer from limited output rates, causing scale ambiguity in monocular setups and possibly a lack of robustness in case of motion blur or illumination changes, for example. On the other hand, IMUs are robust to environmental changes with high sampling rates, but they provide only proprioceptive measurement and suffer from sensors' biases that degrade the acceleration and angular velocity records. With the advent of smart devices embedding both an IMU and a camera, many applications oriented towards the general public adopt viSLAM algorithms. They are a promising alternative that combines multiple sources to increase the tracking quality [20].

3.2. Filter-Based and Keyframe-Based Approaches. As shown in Figure 1, vSLAM methods use two main designs. The first design corresponds to filter-based solutions, which are similar to those that were first used to solve the SLAM problem. This category contains the following:

(i) Extended Kalman filter- (EKF-) based algorithms such as MonoSLAM [21]

(ii) Particle filter-based methods such as FastSLAM and its monocular SLAM alternative [22, 23]

(iii) Multistate constraint Kalman filter- (MSCKF-) based methods such as MSCKF 2.0 [24] or SMSCKF [17]

Classically, filter-based approaches estimate both the camera's pose and the landmarks' positions in a state vector, which is a potential source of scalability inefficiency. MSCKF by Mourikis and Roumeliotis [25] and recent EKF-based VIO solutions, such as ROVIO [26], use a restrictive culling of landmarks to only keep the most recently detected features in the state vector. This local approach to the problem is common for pure visual odometry methods (VO or VIO). It is also possible to keep the 3D features extracted from the state vector in a static map. This map is considered static since it remains unaltered throughout the SLAM process. The mapping is performed after the localization instead of simultaneously.

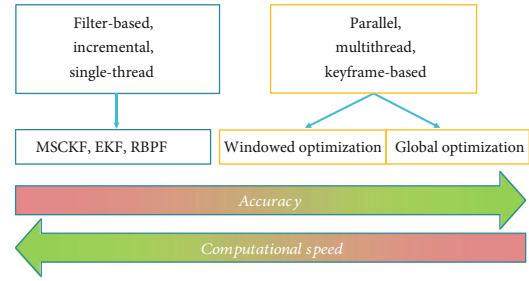


Figure 1: Main design choices for the current vSLAM algorithm.

The second design utilizes parallel methods derived from PTAM [27]. These methods are based on keyframes. The features are parameterized with respect to a keyframe enabling to run in parallel different SLAM tasks on multiple threads. Keyframe-based approaches are also sometimes called "optimization-based" approaches. One of their main advantages regarding performance is that they use a global optimization process bundle adjustment (BA) instead of letting a filter manage the map and poses. Global optimization leads to an improved accuracy thanks to their ability to correct drift effects. However, they are computationally expensive, which is why they were mostly used offline before PTAM [27] introduced the possibility to parallelize the various SLAM's tasks. Just like filter-based methods have evolved into windowed methods to improve computational efficiency and scalability, keyframe-based optimization can only be done on a window of keyframes. Keyframes can also be arranged in graphs. Depending on their design, it is possible to use different criteria to define the windows, instead of using a simple temporal window of the n last keyframes. Optimizations differ also depending on optimization criteria. Pose graph optimization (or motion-only BA) focuses on the poses between keyframes whereas structure-only BA optimizes only the map and BA manages both the map and poses. However, it should be noted that vSLAM methods can also combine various types of optimization designs: e.g., global BA on loop closure and local pose graph optimization at every new keyframe. As summed up in Figure 1, the choice of design is often driven by a compromise between accuracy and speed.

4. Classical Structure of the vSLAM Algorithm

Four main blocks (Figure 2) describe the overall operation of any vSLAM algorithm. They are the following:

(i) Input search: finding the required information in the sensor measurements

(ii) Pose tracking: determining the current camera pose from the new perceptions

(iii) Mapping: adding a landmark to the map

(iv) Loop closing: producing a proper map and drift-free localization

They are detailed in the following subsections.

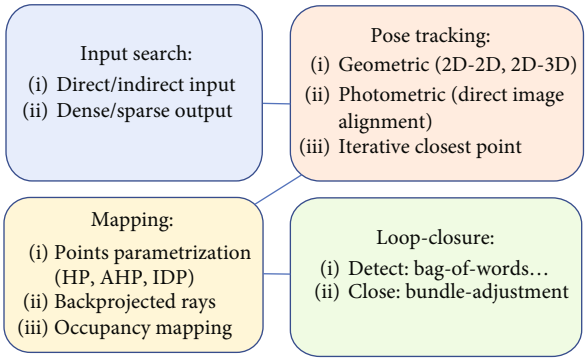


Figure 2: The four main blocks of a vSLAM algorithm, associated with related keywords.

4.1. Input Search. When dealing with cameras, the data contained in frames must first be extracted. Some methods use the pixel intensity to match different frames: they are called direct methods. In this case, the mapped elements can be pixel maps, i.e., the frame is relocated in the 3D map and each pixel is given its corresponding depth (Section 4.3). Other methods extract features (points in the zone of interest of the image, i.e., easily recognizable, or, alternatively, lines or curve segments) in each frame and use geometric constraints for matching. Feature extraction is a well-known field of computer vision. Feature descriptors often use intensity gradients to detect zones of interest. In this case, the mapped elements can be 3D poses of features. Famous descriptors include Harris [28], SURF [29], SIFT [30], FAST [31], and ORB [32]. The choice is usually driven by a trade-off between robustness and computational efficiency. These last methods are called indirect or feature based.

Whereas the choice between direct and indirect is linked to the input space, the choice between dense and sparse maps depends on the output space. The built map is classified from sparse to dense. Sparse maps only contain a cloud of sparse features (i.e., only a small selected subset of the pixels in an image frame) and are typical of vSLAM methods focusing on the correctness of the trajectory. Dense methods use all frames' information (i.e., most or all of the pixel information in each received frame) to reconstruct maps with as much as possible environmental data. Semidense methods are in between. They are dense methods where only specific zones of interest (high-density gradients: edges, corners, …) are mapped. The four input/output combinations that can be imagined based on this analysis all have pros and cons. Table 1 sums the possible combinations giving an example of algorithms that use them.

The choice between direct and indirect is still debated. Feature descriptors are robust to image noise. Recent descriptors can be robust or even invariant to geometric distortions, e.g., due to rolling shutter, automatic exposure changes, and lens vignetting. Direct methods allow using all the information contained in every frame. They avoid the extraction of features and gather map data more quickly. When using a basic camera, such as a webcam or a smartphone, an indirect method will be preferred for its robustness. However, with well-parametrized global shutter cameras, direct methods may be preferred. Dense methods produce more interesting 3D reconstructions of the environment, but they often require heavy parallelization on a high-end GPU.

Table 1: Illustration of algorithms classified according to their input and output methods for the first vSLAM block.

vSLAM	Dense	Sparse
Direct	DTAM [34]	DSO [33]
	LSD-SLAM [35]	ROVIO [26]
Indirect	Optical flow vSLAM [36]	MonoSLAM [21]
		PTAM [27]
		ORB-SLAM [37]

Indirect/sparse methods are the most common methods. They extract features in frames and add them as 3D points for mapping once their coordinates are determined. Direct/-dense (or semidense) methods are pretty common too. Observed pixels are represented by a 3D point in the image frame, constituting a depth map of all pixels. To keep the internal coherence observed in each frame during the mapping, geometric priors are added. They constrain the positions of points, seen from the same frame, using assumptions on the geometry of the scene. Direct/sparse methods are rather uncommon and use photometric error minimization without *a priori* data to keep the correlations between geometry parameters lighter and optimization less time-consuming as in DSO [33]. Indirect/dense methods are rare and do not use feature descriptors. Instead, they compute geometric errors as a deviation from the observed optical flow field in the frame.

4.2. Pose Tracking. The pose tracking block comprises the visual odometry parts. Depending on whether the feature correspondences in two successive frames are in 2D or 3D, there are three different ways to perform visual odometry [11].

(i) 2D-2D alignment: the feature matching is done directly between 2D features detected on successive frames. This is the common solution for pure VO methods

(ii) 2D-3D (sometimes called 2.5D) alignment: the pose of the camera is estimated given a set of n 3D points in the world (mapped points from the precedent frames) and their corresponding 2D projections in the new frame. This problem is known as Perspective-n-Points. Monocular pureVO methods can only use 2.5D alignment by keeping a pool of n frames, triangulating features on the fly, and reusing the 3D poses of these triangulated points before they are culled out by new incoming data. On the contrary, vSLAM benefits from a complete map of landmarks to choose from for reprojection, so 2.5D alignment is commonly used in vSLAM

(iii) 3D-3D alignment: with stereo cameras, it is possible to directly determine the 3D position of a newly detected feature, so 3D-3D alignment can be

considered too. However, 3D feature position estimation generally yield uncertainties bigger than 3D-2D reprojection errors, which is why this method is rarely used

For monocular EKF-SLAM, the reprojection of mapped points is used for the correction phase [38]. MSCKF follows the detected features along a "path," constituted of every detection of the feature from its first detection to the moment it gets out of sight. Then, it performs Gauss-Newton minimization using all features whose path is complete [24]. For parallel methods, VO can be done between every frame while map reprojection can be used to improve the accuracy of a new keyframe's pose estimate. The prerequisite to calculating the new camera pose is to match features between the current frame and a previous one. "Bottom-up" approaches (trying to match features by testing all possibilities) have been replaced by much more efficient "top-down" methods since 2003 MonoSLAM [21]. The basic idea is to compute uncertainty ellipses around the previous frame's features in which a new observation of that feature is supposed to be and to restrict the search space in this ellipse on the new frame. Tracing this ellipse requires some assumptions on the camera movement between the two frames, for instance, CVCAV motion model (Constant Velocity, Constant Angular Velocity) [21]. The solution can usually be found by determining the transformation that minimizes the reprojection error of the triangulated points in each image. With 2D-3D alignment, we search for the transformation that minimizes the reprojection error of the 3D landmarks into the new 2D frame.

2D-2D alignment with features can be done with geometric parameters by computing the essential matrix with epipolar geometry (5-point or 8-point algorithms) and decomposing it into a translation vector $\mathbf{t}$ and a rotation matrix $\mathbf{R}$ (using a singular value decomposition) that form the frame-to-frame transformation $\mathbf{T}$. In direct methods, given a point p, whose pixel coordinates are x in the image I_1 and x' in I_2, $x'(\mathbf{T}, x)$ being the function of the motion $\mathbf{T}$, the true motion $\mathbf{T}^*$ should minimize the photometric difference $I_1(x) - I_2(x')$. Therefore, the best estimate for $\mathbf{T}^*$ is found by minimizing the overall photometric difference in the image, which is a sum of pixel-wise photometric errors (potentially weighted in certain regions, e.g., with high gradients). This is called Direct Image Alignment (DIA) and is the equivalent of 2D-2D alignment for direct methods.

When dealing with a dense model, the tracking is generally done with Iterative Closest Point (ICP) [39–41] using the current frame's depth map and the 3D dense model. Note that tracking based on dense model reprojection can benefit from predictive capabilities improving robustness to occlusion, as well as robust handling of motion blur for instance.

4.3. Mapping. The mapping block refers to the actions required to fully initialize a newly detected feature's position so that it can be situated in the 3D reconstruction of the environment, i.e., the map. Indeed, in the pose tracking section, we assumed the existence of 3D positioned landmarks on our map. However, a monocular camera cannot determine the depth of a feature using only one observation but it needs several frames. Similar to camera trajectory, we may want to estimate the uncertainty of the positions of the landmarks to refine them after their initialization or to include them in global optimization. It is also interesting to use landmarks, which are only "partially initialized," since they already hold some information, such as in PTAM [27]. Direct methods do not map features *per se*, but they map each pixel of the captured frame (pixel depth maps).

The mapping block can be described with the 3D landmark parameterization. Cartesian coordinates (XYZ) can be used, but this choice results in severely non-Gaussian probability density functions and it degrades both the accuracy and consistency. Alternatively, homogeneous point (HP), anchored homogeneous point (AHP), and inverse-depth parameterization (IDP) suppress nonlinearity and shorten the initialization's period [42].

As stated in [42], the IDP encodes the inverse-distance point p by a vector of dimension 6 that contains the Euclidean optical center at the initialization time corresponding to the "anchor point" $p_0 = (x_0, y_0, z_0)$, the elevation and azimuth angles, which define the direction of the initial optical ray (ϵ, α) and the inverse ρ of the Euclidean distance d from p_0 to the 3D point p. IDP points could be parameterized with direct encoding of the optical ray's direction with a vector $V = (u, v, w)$ and the distance $\rho = \|v\|/d$, avoiding the need for nonlinear transformations with the angles (ε, α). This corresponds to an AHP parameterization (7 parameters). HP is similar to AHP but does not need an anchor point; instead, the origin of the camera is used, leading to only 4 parameters V and ρ. Assuming that the uncertainties on the camera position are small, similar results are obtained with this parameterization. The study in [42] details these parameterizations and gives a benchmark of their impact on EKF-SLAM results.

Triangulated 3D points are determined by intersecting back-projected rays from 2D image correspondences of at least two frames. In reality, they never really intersect, which leads to an uncertainty region in the landmark's position. It is possible to reduce this uncertainty by two means. Either more observations can be used or rays from more distant views can improve the positioning. Ideally, the rays should intersect at 90° angle to reach a small uncertainty circle instead of a stretched ellipse. One may skip frames until the average uncertainty of the 3D points' positions decreases below a given threshold to mitigate this issue. The selected frames usually correspond to keyframes. Also, note that distant points are more difficult to map accurately.

The mapping process is a bit different for direct methods such as DTAM [34]. The goal is to turn the captured frames into "depth maps" by assigning a depth value to every pixel. It is again based on multiple-view reconstruction. For each pixel in a keyframe, a pixel ray is traced, which corresponds to the range of possible depths for that pixel. A pool of all frames overlapping with this keyframe is used to "observe" the pixel rays. An energy function minimization, such as L_1 norms sum of photometric errors and prior data for spatial regularization, estimates the real pixel depth. The mapping process is more direct for RGB-D vSLAM methods since

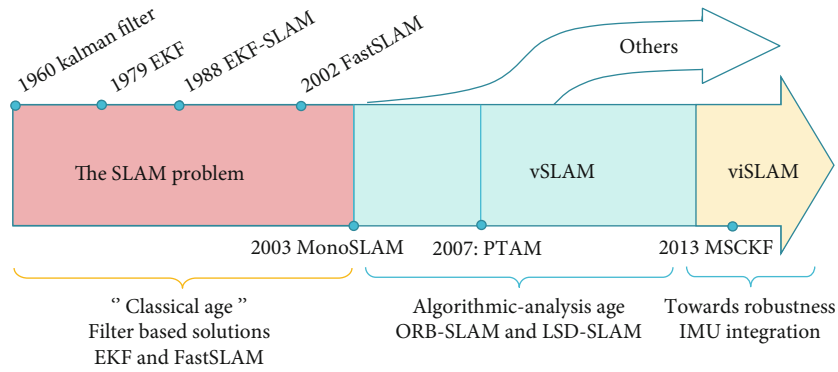


Figure 3: Overview of vSLAM history with milestones and the three ages: from the SLAM problem to vSLAM, vSLAM algorithmic development, and the emergence of viSLAM.

the depth is directly sensed. The depth map is input in each frame. The 3D model of the environment is then elaborated by fusing all depth maps. This can be done either naively by overlapping the scans or by performing fusion methods such as occupancy mapping.

A lot of SLAM methods map the scene with a sparse representation that corresponds to features detected in the environment. Dense maps are more common with stereo and RGB-D camera or laser scanning. A recent work [43] creates a dense map modeling of the dense structure as a Euclidean signed distance field.

4.4. Loop Closure. The loop closure is the backbone of SLAM. It removes the drift accumulated since the last loop closure by reconnecting the pose of a previously visited place with the current pose. Optimization and incremental methods are more successful than particle- and Kalman-based filters because they propagate backward the loop closure data over the trajectory estimate. A keyframe graph is classically used to correct the poses using bundle adjustment (BA) in parallel methods. In [44], the authors synthesize BA's techniques for vSLAM methods and give in an appendix a historical overview of the main developments in bundle adjustment. State-of-the-art SLAM systems are commonly used [45] to solve nonlinear least squares problems or [46] to optimize graph-based nonlinear error functions in BA. But these systems use a few of the last measurements to estimate the pose in real time. To use previous optimizations and reduce computation, incremental solvers, such as [47], solve a nonlinear estimation problem in a real-time approach. They update the estimated model of the environment every time a new measurement is added using the sparse structure of the underlying factor graph.

Loop closing is a two-step process. Firstly, it starts with the loop detection, also called place recognition. The place recognition process can be used to solve the problem of track loss recovery, generally by using the loop closing thread. Most of the methods use a bag-of-word approach to compare new keyframes with a database of previously acquired views. "Bag-of-word" refers to the set of descriptors that identifies patches in images as in the DBoW2 method proposed by [48] based on FAST [31] and a slightly modified version of BRIEF features [49]. A catalog search of similar words between the frame and the database is extremely quick and efficient. Once a potential similarity is found, multiple verification steps verify if it corresponds to a loop. Secondly, the loop closure corrects the map and poses. The transformation between the two views is computed and 3D points are fused. The computations needed to close the loop are then distributed along the entire pose graph and the map using (local) BA. The loop closing process can be computationally heavy. It is generally done in a dedicated thread.

5. Historical Review of vSLAM Methods

Figure 3 shows the chronology of vSLAM's development that comprises three ages. The first age, labeled here the "classical age," focuses on solving the SLAM problem. Several mathematical formulations were proposed, and SLAM was effectively applied for the first time. During the "second age," the focus of SLAM research moved to vision-based approaches. Several vSLAM designs were proposed and new hardware, such as GPU, RGB-D cameras, and stereo cameras, were integrated into the process. This "age of vSLAM" concurs with what [14] refers to as the "algorithmic analysis" age of SLAM. Fundamental properties of vSLAM were studied, such as convergence and consistency. vSLAM became central in the development of SLAM methods. The "third age" is dedicated to improving the robustness of vSLAM. The goal was to improve vSLAM reliability to support the increasing number of real-life applications (e.g., drones). In particular, this "third age" introduced viSLAM approaches.

5.1. The Classical Age. The recent history of localization started with the introduction of the Kalman filter in 1960 [50], extended to nonlinear systems by Maybeck in 1979 with the extended Kalman filter (EKF) [51]. The SLAM problem was formulated in the 1980s [1, 2, 52] and proved convergent in 1995 [53]. During this period, a few SLAM approaches were formulated, mainly with laser telemeters, odometry calculated from different sources and implemented with EKF, such as the one proposed as early as 1988 by Smith et al. (the EKF-SLAM [38]). The use of a monocular camera was very rare until 2003 when Davison et al. proposed MonoSLAM [21]. They implemented it using only one webcam, a generic computer, and without odometry measurements. It

was the first real-time SLAM method using a single low-cost visual sensor. Mapping and localization were performed in 3D, and the SLAM was based on an EKF. To deal with the problem of initializing new points, Davison et al. proposed a new method based on a particle filter to reduce the uncertainty on field depth for newly detected visual landmarks. MonoSLAM paved the way for what will be known as vSLAM.

Every EKF-SLAM, even the famous MonoSLAM, suffered from complexity, quadratic in the number of mapped features. Many attempts were made to mitigate the problem, especially by dividing the map into parts and using only the active submap during the optimization process. None of them provided a satisfactory consistency versus computational cost compromise. In 2002, a Rao-Blackwellized particle filter was used instead of an EKF in the FastSLAM proposed by Montemerlo et al. [23]. This method effectively reduced the complexity of logarithmic scaling, with a successful transposition into monocular vSLAM, i.e., scalable monocular SLAM by Eade and Drummond in 2006 [22].

5.2. The Golden Age of vSLAM. Even the smallest complexity of FastSLAM methods severely limited SLAM applications, especially vSLAM that captures a lot of features. The biggest breakthrough in vSLAM was the introduction of keyframe-based solutions with Parallel Tracking and Mapping (PTAM) by Klein et al. in 2007 [27]. Among other improvements, this new approach enabled task parallelization, better use of global optimization, a reduced tracking drift, and more importantly a new way of storing features with free scalability. Almost every vSLAM algorithm is based on PTAM's concept, nowadays.

vSLAM became increasingly reliable with the integration of efficient loop closure methods, global optimization, and memory management based on keyframes and culling, with processes performed in real time thanks to the multithread parallelization. Consequently, new design and hardware choices could be proposed broadening SLAM possibilities. The development of vSLAM driven by use case requirements became possible. Main vSLAM implementations proposed during this period are detailed in Section 6.

5.3. The Third Age: Improving the Robustness. During the third age, vSLAM continued to evolve, especially to improve robustness targeting specific scenarios. The coupling of cameras and IMUs (viSLAM) became an important research topic. In the early 2010s, loose coupling of IMU data in existing vSLAM methods was considered [54]. But hybridization filters rapidly evolved to the design of "tightly coupled" visual-inertial methods, which are now very popular for systems equipped with IMU and camera. An important improvement in tightly coupled viSLAM is the 2007 MSCKF (multistate constraint Kalman filter) by Mourikis and Roumeliotis [25], improved with MSCKF 2.0 in 2013 [24] that introduced a new version of the Kalman filter that combines observations overtime in one exteroceptive update. Other remarkable methods were also created. Among them is the use of lines instead of point features to avoid motion blur-related issues, which is still used since PTAM. Other works use RGB-D camera and filter the data associated with moving objects before applying the SLAM algorithm to improve robustness [55, 56]. Other approaches mitigate the dynamic problem using an optical flow-based approach to detect and discard dynamic features [57]. A new field of research, combining SLAM and deep learning techniques, is emerging on this topic. Deep learning techniques detect moving objects and support ORB-SLAM2 algorithms to construct an accurate map and localize moving robots in a dynamic environment [58]. In [59], the authors use semantic segmentation and RGB-D camera for the same purpose. Rosinol et al. and Yang et al. [60, 61] publish recent works using semantic segmentation and SLAM algorithm.

When it is possible to revisit several times the same place, the MapLab [62] work allows merging different maps of different sessions at a large scale. The output map can be used from one session to another.

Perhaps one of the most interesting recent trends is the use of event cameras, i.e., bioinspired cameras, to avoid the effect of motion blur. Because the use of event cameras is still very recent, it is not included in our classification. However, results published in 2018 by Rosinol et al. about Ultimate SLAM [63] mixing the use of an event-based camera and visual-inertial odometry seem very promising and eager to open up new possibilities for vSLAM.

A recent work [64] optimizing both local and global bundle adjustments gives promising results that can be used in the viSLAM algorithm to improve global consistency.

6. Proposed Classification of Methods

Several approaches drive our classification's work. Section 6.1 groups vSLAM methods based on their inputs. Section 6.2 groups viSLAM methods based on the level of coupling. Finally, Section 6.3 compares the main v/viSLAM methods according to the hardware requirements, the algorithm types, and the implementation features. It is completed by a cross-analysis of the v/viSLAM performance depending on the application requirements.

6.1. Classification of vSLAM Methods. vSLAM methods belong to three categories depending on the nature of the input: feature based, direct, and RGB-D based, as introduced in Section 4. Because RGB-D-based methods involve specific hardware, it is considered as a whole category. Figure 4 shows the classification result with a selection of some of the main identified methods.

6.1.1. Feature-Based Methods. The monocular EKF-SLAM MonoSLAM [21] and the particle filter scalable monocular SLAM [22], i.e., monocular FastSLAM, belong to the feature-based methods. The breakthrough Parallel Tracking and Mapping (PTAM) [27] belongs to the same category. Several adaptations of PTAM were proposed. The use of edgelet features was, for example, introduced in [65]. Another important method is the ORB-SLAM [37].

6.1.2. Direct Methods. The first important direct method is the Dense Tracking and Mapping (DTAM) [34] from 2011. It was a pioneer of dense monocular vSLAM methods and adapted for smartphones in 2015 with the MobileFusion

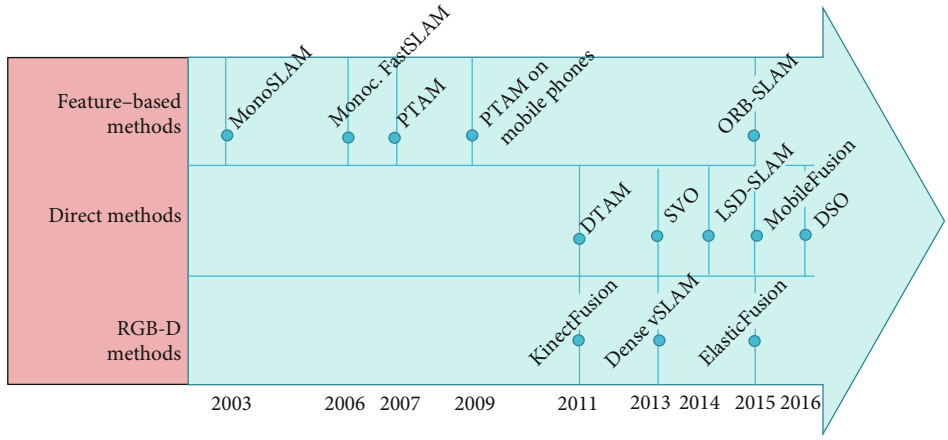


Figure 4: Chronological classification of the main vSLAM methods.

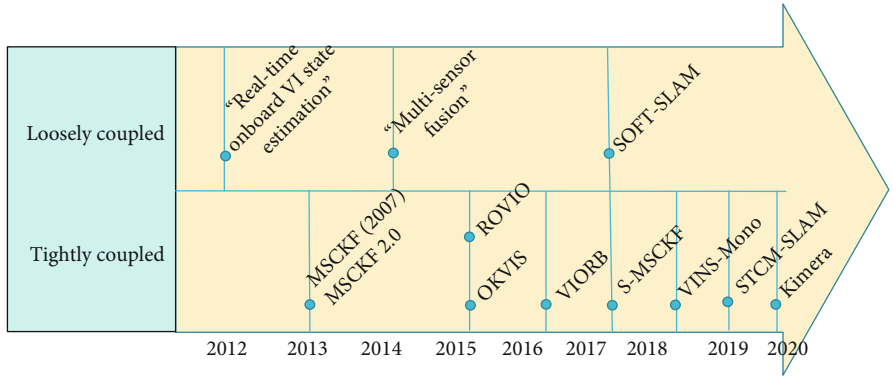


Figure 5: Chronological classification of main visual-inertial SLAM methods.

[66]. A more recent method, from 2016, is the Direct Sparse Odometry (DSO) [33]: a visual odometry method that proposes direct input treatment but sparse mapping for lighter processing. Another major vSLAM method is the Semidirect Visual Odometry (SVO) in 2013 and 2017 [67] that combines the advantages of both direct and indirect input searches in a VO framework. Large-scale direct monocular SLAM (LSD-SLAM) in 2014 [35] is one of the first methods that uses semidense mapping to address large environments.

6.1.3. RGB-D Methods. RGB-D methods include also several main algorithms. In 2011, the KinectFusion [68] was aimed at building a clean and accurate 3D reconstruction of an environment using the Microsoft Kinect. Dense vSLAM [69], from 2013, focuses on accurate localization taking advantage of dense maps. The ElasticFusion [70], in 2015, is a "map-centric" method that focuses more on the geometric accuracy of the built 3D model than on the construction of a pose graph.

6.2. Classification of viSLAM Methods. Direct and indirect features could be used to classify viSLAM methods. Other reviews have also classified viSLAM methods depending on whether they are filter- or optimization-based methods [15]. But most major viSLAM methods are actually feature-based methods and viSLAM mainly deals with hybridization issues. Therefore, the classification shown in Figure 5 is based on the coupling level of the visual and inertial data. We differentiate two levels: loose and tight coupling.

6.2.1. Loose Coupling. Loosely coupled methods process the IMU and image measurements separately and use both information to track the pose. Weiss et al. [54] process images to compute VO between consecutive poses and subsequently fuse the latter with inertial measurements. IMU measurements can also be filtered to estimate rotations that are fused in an image-based estimation algorithm. Loosely coupled visual-inertial odometry method is one part of the global multisensor fusion (magnetometers, pressure altimeters, GPS receiver, laser scanners, …) addressed by [71] in 2014. Although the interest for visual-inertial systems is quite recent, works on loosely coupled IMU-camera fusion started already in the early 2000s. SOFT-SLAM algorithm [72] is a loosely coupled viSLAM method that in fact uses IMU data to reduce computation time when available. It builds in real time a dense map and runs on a MAV.

6.2.2. Tight Coupling. Instead of fusing the outputs of vision- and inertial-based algorithms, tightly coupled methods fuse directly visual and inertial raw data to improve accuracy and robustness. The MSCKF [25] and MSCKF 2.0 [24], both robust and very light, belong to this category, along with ROVIO [26], which is an EKF-based direct VIO method.

Table 2: Comparative classification of main vSLAM and viSLAM methods. The algorithms adapted to pedestrian navigation applications are presented in bold.

Algorithm map gestion	Hardware requirements				Approach			Input treatment		Localis./Mapping			Memory loop	
	Monoc.	Stereo	Depth	IMU	Filter	Optim.		Direct	Indir.	2D-2D	3D-2D	IMU	closure	
MonoSLAM [21]	X				X		Sparse		X		X			
Monocular FastSLAM [22]	X				X		Sparse		X		X			
PTAM [27]	X					X	Sparse		X		X			
PTAM with edgelets [65]	X					X	Sparse		X		X			
PTAM with DWO [79]	X					X	Sparse		X		X			X
Stereo PTAM [78]		X				X	Sparse		X	X	X			X
CD-SLAM [80]	X					X	Sparse		X		X			X
ORB-SLAM [37]	X					X	Sparse		X	X	X		X	X
ORB-SLAM2 [76]	X	(X)	(X)			X	Sparse		X	X	X		X	X
Edge-SLAM [81]	X					X	Sparse		X	X	X			X
DTAM [34]	X					X	Dense	X		(X)	X			
MobileFusion [66]	X			(X)		X	Dense	X			X	(X)	X	
Semidense visual odom. [5]	X					X	Semidense	X		X			X	
LSD-SLAM [35]	*X*					*X*	*Semidense*	*X*		*X*				*X*
Semidirect VO (SVO) [67]	X					X	Sparse	X	X	X	X		X	
Direct sparse odom. (DSO) [33]	*X*					*X*	*Sparse*	*X*		*X*			*X*	
KinectFusion [68]			X			X	Dense	X			X			
Kintinuous [82]	(X)		X			X	Dense	X			X			X
DVO SLAM [69]	X		X			X	Dense	X		X				X
ElasticFusion [70]	X		X			X	Dense	X			X		X	X
MSCKF [25]	X			X	X		None		X		X	X	X	
MSCKF 2.0 [45]	X			X	X		None		X		X	X	X	
ROVIO [26]	X	(X)		X	X		None	X	X	X		X	X	
OKVIS [73]	(X)	X		X		X	Sparse		X	X	X	X	X	
S-MSCKF [17]		X		X	X		None		X		X	X	X	
Vins-Mono [74]	X			X	X		Sparse		X		X	X	X	X
Kimera [60]	(X)	X		X	X	X	Dense		X	X	X	X	X	X
SOFT-SLAM [72]		X		(X)		X	Dense		X	X	X	(X)	X	X
STCM-SLAM [77]		X		X		X	Sparse		X		X	X	X	X
VIORB [75]	X			X		X	Sparse		X	X	X		X	X

Open Keyframe-Based Visual Inertial System (OKVIS) [73] and S-MSCKF [17] are famous stereo VIO methods, while Vins-Mono [74] is a real viSLAM and not just a VIO method. Kimera [60] is also based on a VIO method but it also includes a pose graph optimizer, in different threads, for global trajectory estimation, a 3D mesh reconstruction module, and a 3D metric-semantic reconstruction module. VIORB [75] is based on ORB-SLAM [76]. Its front-end extracts feature with ORB while its back-end runs graph optimization. But its main interest lies in a new IMU initialization method that first estimates the gyroscope's bias, approximates the scale and the gravity (without considering accelerometer bias), and then estimates the accelerometer bias (with scale and gravity direction refinement) and finally the velocity vector. It includes global optimization and loop closure in parallel methods. Most of the recent viSLAM methods are tightly coupled [15], as the one presented by [77] that uses forward and backward optical flow to tack image features.

6.3. Comparison of vSLAM and viSLAM from the Usage Point of View. Main v/viSLAM methods are compared in Table 2 according to the hardware requirements, the algorithm types, and the implementation features. Table 3 presents the state-of-the-art performance of each method for five key features describing the nature of common use cases. Table 3 has been compiled by reading the cited publications and using the criteria below to classify their performance evaluation results.

(i) **Lifelong experiment** feature assesses how the method deals with lifelong experiments. Even in

Table 3: Indications on the robustness to various scenarios of the most famous vSLAM methods.

Algorithm	Recommended usages					Objectives
	Lifelong exp.	Large envir.	Low textured	Outdoor (light, outliers)	Robust to mov.	
MonoSLAM [21]	−	−	−	−	−	Pose estimation in robotics
PTAM [27]	−	−	−	−	~	A.R. in small workspaces
ORB-SLAM2 [76]	+	+	~	+	+	Robust large path tracking
Edge-SLAM [81]	~	+	+	+	+	Low-textured environments
DTAM [34]	−?	−	~	~	+	Robustness to motion blur
MobileFusion [66]	+	−	−	−	−	3D object modeling on phone
LSD-SLAM [35]	~	+	−	+	~	Semidense trajectory estimation
SVO [67]	+	~	+	+	+	Fast, consistent, semidirect method
DSO [33]	+	~	+	+	+	Direct and sparse VO method
KinectFusion [68]	+	−	+	+	~	3D modeling with the Kinect
ElasticFusion [70]	−	−	+	~	~	Map-centric vSLAM
S-MSCKF [17]	+	~	~	~	+	Rapid and consistent Kalman filter
ROVIO [26]	+	~	−	~	~	Robust VIO for UAVs
OKVIS [73]	+	+	~	+	+	Robust stereo VIO for UAVs
Vins-Mono [74]	+	+	~	+	+	Full viSLAM method
Kimera [60]	+	~	+	~	+	VIO+3D semantic-metric mesh
VIORB [75]	+	+	~	+	+	VI method based on ORB-SLAM

For each difficulty, we consider the method to be either robust (+), to have potential difficulties (~), or to not be recommended at all (−). This does not reflect the overall accuracy of the method or the robustness of the initialization procedure.

small environments, vSLAM continuously gathers new keyframes, which means lifelong experiments may lead to an increasing number of keyframes and mapped features. Defining a good keyframe selection policy and memory management methods is important

(ii) **Large-scale environment** feature judges the ability of the method to scale to large environments (i.e., more features accumulated and more distant landmarks). To assess this aspect, important criteria are drift mitigation, efficient global optimization (e.g., choice of keyframes and type of keyframe graph), and efficient place recognition and loop closure frameworks

(iii) **Low-textured environments** correspond to surfaces with few textures such as large walls that can be a problem, especially for feature extraction. Methods with the best performance facing low-textured spaces may use special features like edges and additional hardware for localization (IMU) or, even better, for mapping (depth sensors of RGB-D methods)

(iv) **Outdoor environments**: this type of environment adds difficulties such as light changes. The use of depth sensors, good feature descriptors, or simply robust place recognition methods (to correct errors by closing loops) is relevant for outdoor spaces. The analysis is also based on the outcome of outdoor tests for each algorithm

(v) **Movement:** robustness to the motion blur induced by high-speed movements when dealing with cameras is important to choose the right method for the right application. Methods using a dense map or edge features are generally more robust. However, visual-inertial systems are found to be more robust, as discussed earlier in Section 4

Other methods are added to complete the v/viSLAM classification performed earlier. Two interesting PTAM-derived algorithms are the stereo version of PTAM [78] and the Double-Windowed Optimization (DWO) [79] framework. CD-SLAM [80] was the first attempt to extend PTAM's principles to large-scale indirect vSLAM in a more robust approach. Some of CD-SLAM's features inspired the popular ORB-SLAM. The latter combines the most efficient vSLAM features (bimodel initialization, efficient keyframe and map point culling, ORB features, and bag-of-word loop closure) as well as the "essential graph" of keyframes. ORB-SLAM2 [76] adapts ORB-SLAM for stereo and RGB-D cameras. Edge-SLAM [81] is a recent attempt to build a robust vSLAM with edges as features. The direct method "semidense visual odometry" [5] and Kintinuous [82] are also included. The first one focuses on the application of semidense vSLAM for augmented reality using a smartphone. The last one is an extension of the KinectFusion algorithm for larger-scale environments.

7. Experimental Benchmark

Using the classification in Section 6, five main algorithms were selected for the experimental benchmark: DSO [33], LSD-SLAM [35], ORB-SLAM2 [76], and two viSLAM methods: ROVIO [26] and Vins-Mono [74]. They can all be considered

Table 4: Datasets chosen for our benchmarking.

Dataset	Difficulty	Characteristics
Machine Hall 01	Easy	Low velocity, well textured, good illumination, 80.6 m (182 s)
Machine Hall 03	Medium	High velocity, well textured, good illumination, 130.9 m (132 s)
Machine Hall 05	Difficult	High velocity, poorly textured, bad illumination, 97.6 m (111 s)
IRSTV	Difficult	Low velocity, well textured and poorly textured part, illumination changes (outdoor and indoor parts), 466 m (390 s)

as reference methods in their categories. They embody the variety of existing designs and are theoretically suited for our use case. They are presented in bold in Table 2. Based on our classification, we choose the viSLAM algorithms ROVIO and Vins-Mono for the pedestrian context. They were also considered by [3] as the most accurate and robust algorithms across all platforms and datasets, and ROVIO is considered as a good compromise. Chen et al. [15] also consider Vins-Mono as the algorithm with the best accuracy among the viSLAM algorithms under test.

Not only were these algorithms selected based on their importance in the field but also on the requirements of the use case of interest: pedestrian's pose estimation with a handheld device in urban environments. Selected methods should target accurate pose estimation in a challenging context and be available in open source. No restriction linked to computational difficulties was applied for this benchmark. Global robustness to many perturbations is expected as we seek an implementation on a handheld device, such as a smartphone. We only selected methods that work with a monocular camera and IMU for the viSLAM part. Pedestrian applications mean that the user's behavior has to be considered. The movement of a handheld camera can be shaky, and fast uneven motion will occur. In terms of user requirements, we preferred methods that did not require a very specific manual initialization. As most of the pedestrian applications are being outdoors (e.g., in urban environments), the selected algorithms must handle large-scale scenarios, long experiments (several minutes to an hour), and light changes. Kimera [60] and VIORB [75] could have been added to this benchmark considering their assessment in the above classification. But even if Kimera aims at running with monocular camera, one part of the code (the loop closure detection) requires a stereo camera at the moment. As for VIORB, the authors have not publicly distributed their code.

7.1. Experiments and Dataset Features. The following features were adopted for the experimental setup, data acquisition, and analysis. They are specific to our use case.

(i) Hardware (handheld device): forward-facing monocular camera with rolling shutter and IMU recording

(ii) Pose estimation: our main focus is the online correctness of the current pose, not the overall accuracy of the reconstructed path

(iii) Outdoor scenarios: illumination changes, the variability of the scale of the observed scene

(iv) Pedestrian: few changes in overall velocity (and low mean velocity) but a shaky camera. Accelerations follow a specific back and forth movement on each axis due to the walking pace

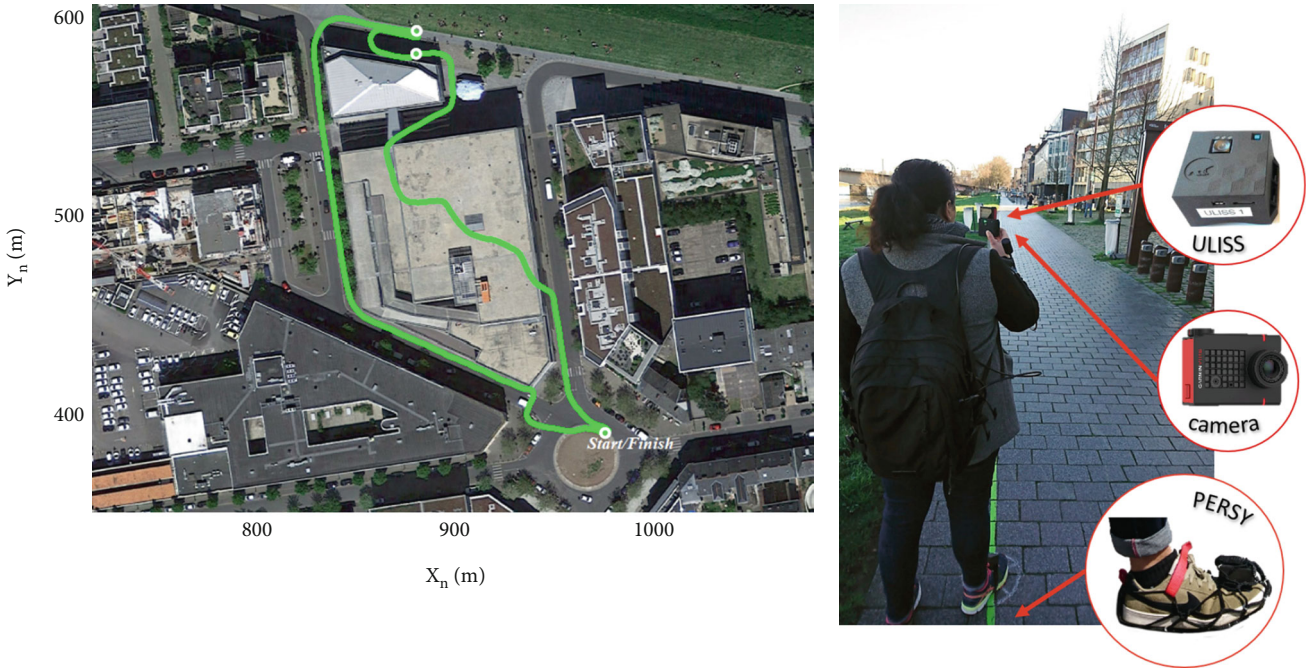
(v) Large trajectories: it expands from a few to no loop closure

(vi) Urban environments: moving objects, potential moments with badly textured elements observed (sky, wall, etc.)

(vii) General public: user-friendliness is expected (no specific manual initialization, no need to adjust parameters to each scenario)

The EuRoC MAV dataset [6], including inertial data, was chosen for the overall comparison of all five methods. Other IMU+vision datasets could have been considered to extend our experimental assessment. In [83], an indoor/outdoor dataset is described but its ground truth accuracy is only 15 cm as compared to the 1 mm with the EuRoc dataset. The dataset presented in [84] is recorded by a MAV in Zurich urban space. The ground truth of the MAV displacement is postprocessed using Pix4D photogrammetry mapping instead of being surveyed by an external localization system (e.g., a motion capture system) as it is in the EuRoc dataset. The dataset presented in [85] is another interesting candidate with good accuracy and higher test sequence variability than EuRoc. We choose EuRoc since its broad adoption in other reviews eased the comparison of our results with other studies. We will consider other datasets in future work. Interested readers can find other datasets referenced in [15, 83–85].

EuRoc comprises various scenarios of drone flight in the same environment and six degrees of freedom ground truth. We choose to focus only on three scenarios among the 11 available. They correspond to different characteristics, detailed in Table 4, which are all interesting to assess the performance of the selected algorithms. Let us note that the drone flight movements show similarities with those of a handheld camera, contrary to the car-embedded cameras of the KITTI dataset [86], whose 10 Hz sampling frequency is not sufficient to track rapid movements. The amplitude of hand movement is often underestimated. Movements are more important than those of the arm naturally synchronized with the walking gait. The handheld device is turned over and raised to explore the environment. This is why we can compare it to the drone dynamics. The selected methods were also tested on a new dataset called "IRSTV" collected by a pedestrian walking with a handheld device along the urban



(a) (b)

Figure 6: (a) IRSTV dataset path and (b) experimental setup.

Table 5: Relative pose error for each tested method on each dataset, averaged on five runs.

RPE (cm)	Vins-Mono $\overline{RMSE}$ ($\sigma\overline{RMSE}$)	ORB-SLAM2 $\overline{RMSE}$ ($\sigma\overline{RMSE}$)	DSO $\overline{RMSE}$ ($\sigma\overline{RMSE}$)	ROVIO $\overline{RMSE}$ ($\sigma\overline{RMSE}$)	LSD-SLAM $\overline{RMSE}$ ($\sigma\overline{RMSE}$)
MH 01 (easy)	2.99 (1.8%)	3.14 (0.9%)	2.98 (0.6%)	9.98 (0.6%)	6.32 (9.2%)
MH 03 (medium)	3.70 (0.8%)	3.19 (1.0%)	3.68 (2.4%)	8.89 (0.0%)	—
MH 05 (difficult)	3.85 (0.9%)	3.73 (0.5%)	3.49 (0.4%)	16.0 (0.0%)	17.9 (9.9%)

path, shown in Figure 6, along with the hardware setup. It corresponds to a 466 m walking path in both indoor and bright outdoor spaces. It comprises images with scenes at different scales: streets, open spaces, and rooms. It also contains observations of glass-covered buildings that reflect the environment, which classically fails visual odometry. No specific motion for dynamic initialization was imposed at the beginning of the acquisition.

The hardware setup comprises a Garmin camera and a dedicated platform named ULISS [87] (Figure 6). The camera is the "VIRB 30 Ultra" with a fixed focal length, a 60 Hz frame rate, and a 1920 × 1080 pixel resolution corresponding to a standard resolution of smartphone's acquisition. ULISS comprises a triaxis inertial measurement unit and a triaxis magnetometer sampled at 200 Hz, a barometer, a high-sensitivity GPS receiver, and an antenna. These low-cost sensors are classically embedded in mobile devices. This hardware setup gives access to raw data without prefiltering often embedded in mobile devices. ULISS and the camera data are precisely synchronized using timestamps from the GPS receivers embedded in both devices. The reference system for indoor/outdoor pedestrian navigation, PERSY [88], was attached to the foot. It provides the ground truth for the pedestrian's scenario with 0.3% horizontal positioning accuracy of the cumulative walking distance. It is shown in green in Figure 6.

The benchmarking is conducted on a 2.60 GHz Intel Core i7-6700HQ CPU. Our Linux environment is in a virtual machine (Oracle VirtualBox 5.2.12). We allowed all four cores with 100% allocated resources and 5 GB of RAM usage. All algorithms are tested in Ubuntu 16.04. When ROS was required, we used ROS kinetic with a catkin workspace. The next section presents a detailed analysis of the chosen algorithms on the selected dataset and assesses the most suitable for our case.

7.2. Experimental Assessment Methodology. We ran the tests on the two previously presented datasets starting with EuRoc dataset. All results are averaged on five runs to account for random outlier mitigation parts. In Tables 5 and 6, we provide the mean values of the Relative Positioning Error (RPE) and the Absolute Positioning Error (APE) for each method on each dataset. APE and RPE wre obtained with obtained with evo package github.com/MichaelGrupp/evo. APE is the Euclidean distance between the estimated position (2D or 3D) and the true position whereas RPE is the

Table 6: Absolute pose error for each tested method on each dataset, averaged on five runs.

APE in cm	Vins-Mono			ORB-SLAM2			DSO			ROVIO			LSD-SLAM		
	$\overline{RMSE}$ ($\sigma\overline{RMSE}$)	$\bar{\sigma}$	$\overline{Max}$	$\overline{RMSE}$ ($\sigma\overline{RMSE}$)	$\bar{\sigma}$	$\overline{Max}$	$\overline{RMSE}$ ($\sigma\overline{RMSE}$)	$\bar{\sigma}$	$\overline{Max}$	$\overline{RMSE}$ ($\sigma\overline{RMSE}$)	$\bar{\sigma}$	$\overline{Max}$	$\overline{RMSE}$ ($\sigma\overline{RMSE}$)	$\bar{\sigma}$	$\overline{Max}$
MH 01	8.46 (11.50%)	3.79	20.3	4.3 (1.29%)	2.00	7.93	7.57 (2.47%)	3.54	16.28	30.4 (0.90%)	15.08	96.96	11.76 (12.45%)	8.27	89.92
MH 03	9.51 (7.45%)	4.71	25.25	3.89 (2.68%)	1.71	9.47	10.05 (7.70%)	4.71	23.90	39.25 (0.00%)	16.44	78.03	—	—	—
MH 05	17.39 (2.71%)	7.51	32.01	5.31 (3.21%)	2.19	11.89	13.87 (4.21%)	5.51	24.30	105.45 (0.09%)	49.41	223.36	*101 (12.1%)*	*59.3*	*722*
IRSTV	695	367	2395	649	396	1657	1116	897	5188						

Results in *italics* indicate that the algorithm failed on some of the five runs by losing track. The numbers are thus average on the other runs. The numbers in parentheses are the standard deviation of the RMSE (which is averaged on five runs). Results on IRSTV's dataset are further explained in the result analysis of each concerned method. None of the methods reconstructed the full IRSTV path but only some parts of it: DSO 292.75 m, ORB-SLAM2 596.80 m, and Vins-Mono 212.83 m. APE was obtained with *evo* package github.com/MichaelGrupp/evo.

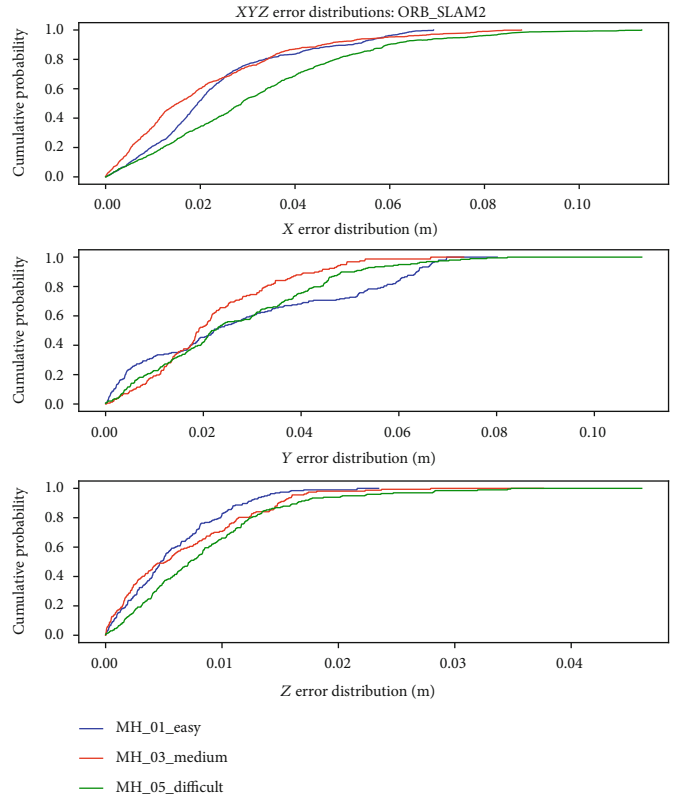


Figure 7: ORB-SLAM2: cumulative error distributions alongside each axis of the position on EuRoC MH01, MH03, and MH05.

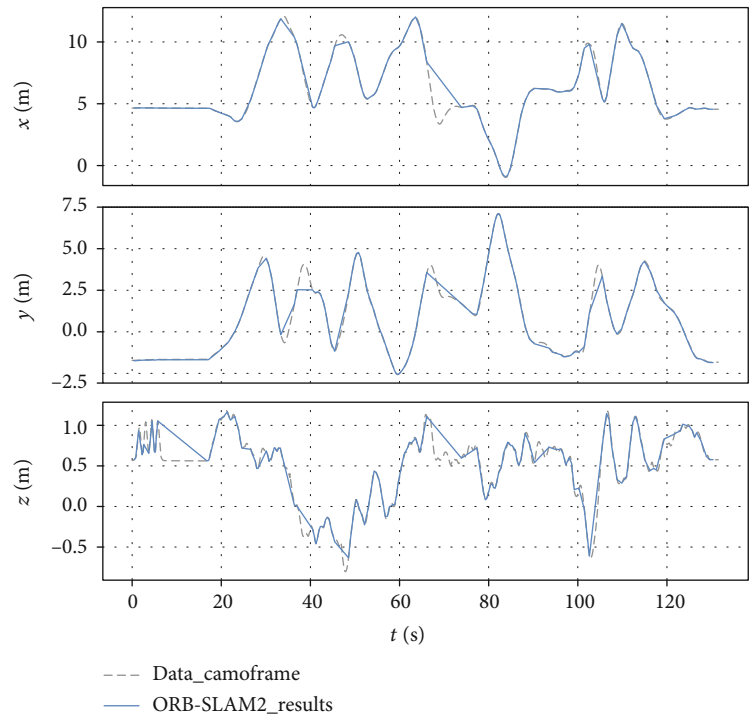


Figure 8: ORB-SLAM2: trajectory plots showing moments where parts of the tracked trajectory are not estimated (at the end). The example is for one run on EuRoC MH03.

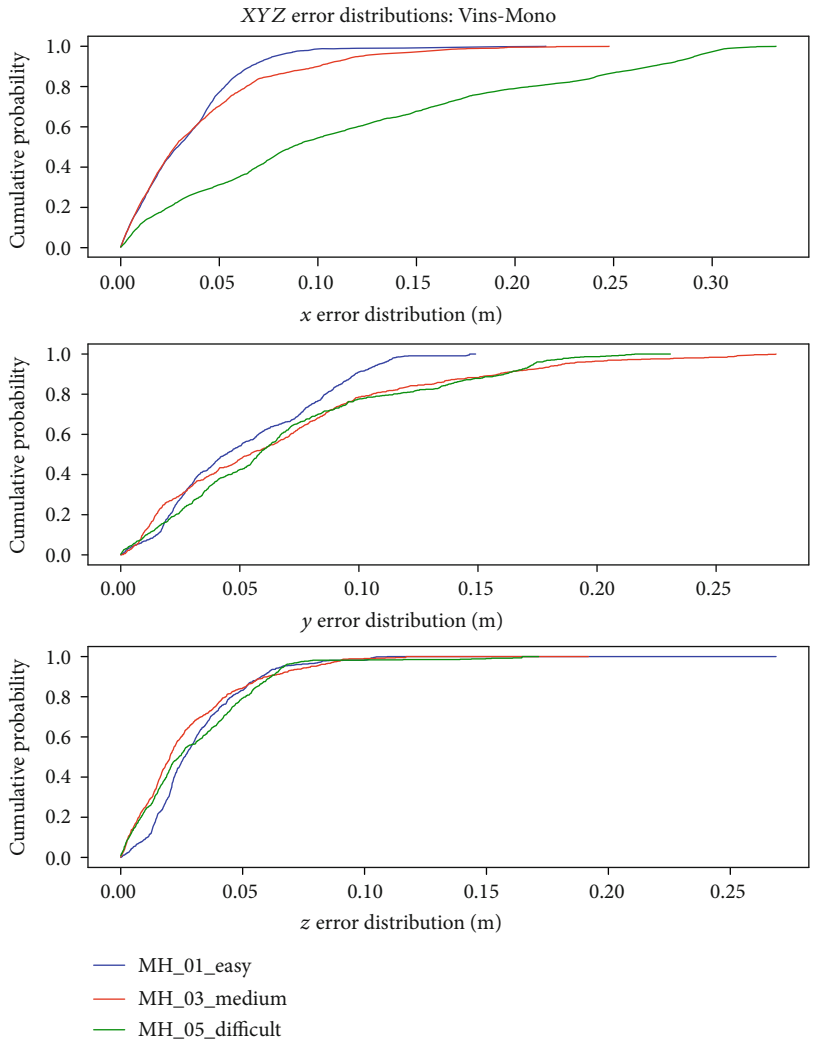


Figure 9: Vins-Mono: cumulative error distributions alongside each axis of the position on EuRoC MH01, MH03, and MH05.

Euclidean distance between consecutive position estimates (2D or 3D). We computed the average values of the root mean square error ($\overline{RMSE}$), of the standard deviation ($\bar{\sigma}$), and of the maximal error ($\overline{max}$) as constancy indicators. The APE is calculated after Sim (3) Umeyama alignment for all methods [89]. The RPE is calculated as a drift: translation or rotation error per meter. This local accuracy is independent of the number of keyframes and allows to compare VO and vSLAM.

RPE assesses the drifting error part by checking the correctness of the pose-pose transforms. For instance, a good RPE and a bad APE may indicate a bad drift correction. It may come from an inefficient loop closure framework and/or bad initialization knowing that initialization is one of the weaknesses of SLAM [20]. It can also mean that major "singular" errors are likely to happen locally and are not corrected. In this case, the max RPE should be much bigger than the RMSE RPE.

Using the provided timestamps, we compared the real position with the estimated position to compute APE. They are computed in 3D for the EuRoc dataset but only in 2D for the IRSTV dataset since PERSY only provides the ground truth of the pedestrian's path in the horizontal plane. The cumulative APE distribution functions (CDF) are plotted for global accuracy and stability assessment of the tested algorithms. The errors are computed in the local navigation frame defined by the x and y axes forming the local horizontal plane and the z axis pointing upward. To simulate real-life applications where no human intervention is needed between various tasks in diverse environments, the parameters needed to run the computation were fixed for all runs.

We also evaluated the performance on the five features, pertinent for the use case (Section 7.1), as well as the stability (whether or not the results are likely to change for a given scenario from one run to another), the overall quality of the pose estimation, and the ability to estimate the true scale without postprocessing. For this last point, we simply determined the relative error over the total length of the trajectory between the results and the ground truth.

$$e_{\text{scale}} = \frac{\left|\text{length}_{\text{result-traj}} - \text{length}_{\text{groundtruth}}\right|}{\text{length}_{\text{groundtruth}}}. \quad (1)$$

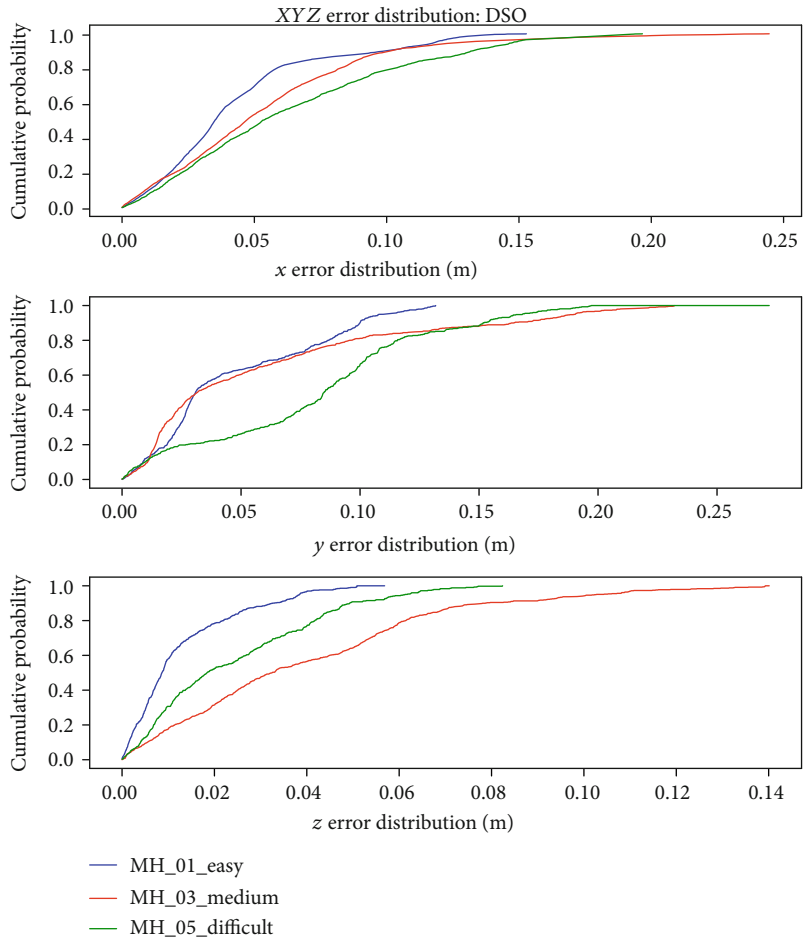


Figure 10: DSO: cumulative error distributions alongside each axis of the position on EuRoC MH01, MH03, and MH05.

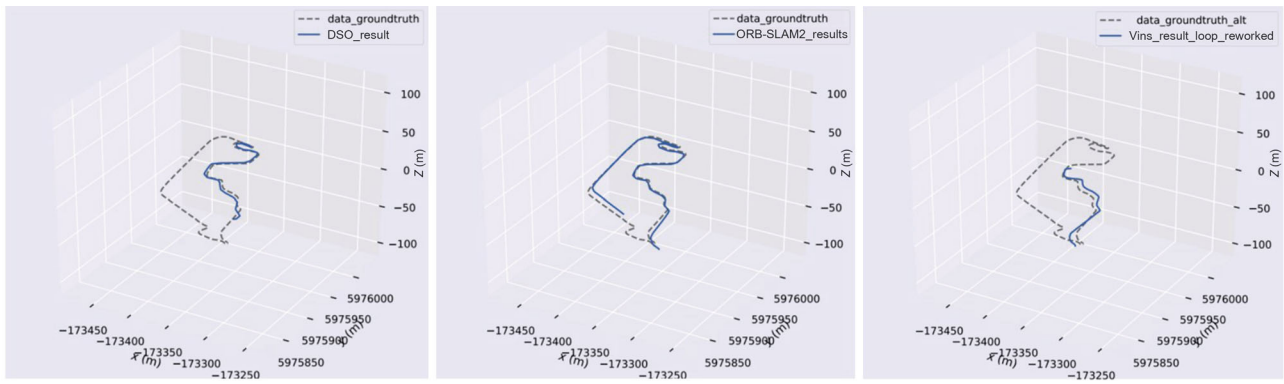


Figure 11: Trajectories effectively traveled on IRSTV dataset. From left to right: DSO (292.75 m), ORB-SLAM2 (596.80 m), and Vins-Mono (212.83 m).

7.3. Detailed Analysis. The analysis starts with monocular ORB-SLAM2, Vins-Mono, and DSO giving the best RPE and APE on the three EuRoc. It continues with LSD-SLAM and ROVIO providing less good results and failing on the IRSTV dataset.

7.3.1. ORB-SLAM2 Result Analysis. Globally, ORB-SLAM2 provides the best RPE RMSE. It outperforms DSO and Vins-Mono in terms of APE, both at the RMSE and the maximum errors. This highlights its efficiency to correct the drifting effect. Figure 7 shows the CDF of the positioning error for the 3 EuRoc. We observe a very good constancy with the lowest σ of this benchmark and the best predictability after ROVIO with the lowest RMSE's standard deviation. ORB-SLAM2 gives noticeable constancy with a max APE/RMSE ratio of around 2, the lowest σ/RMSE ratio of this benchmark. Apart from ROVIO, it gives the best predictability with the lowest RMSE's standard deviation.

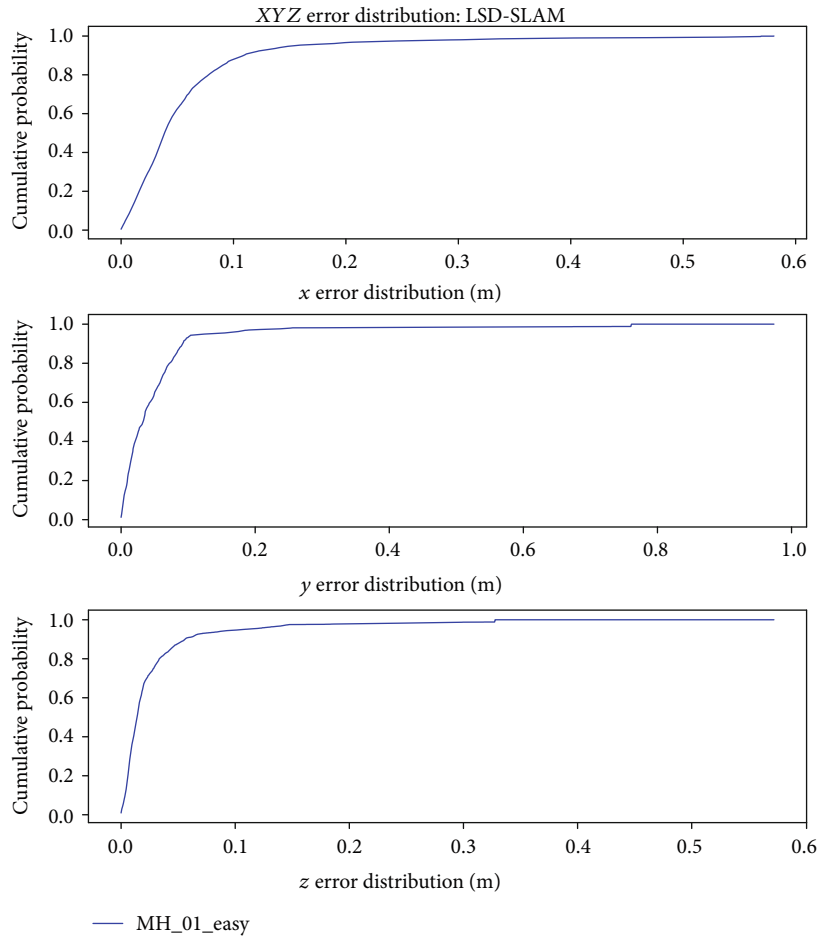


Figure 12: LSD-SLAM: cumulative error distributions alongside each axis of the position on EuRoC MH01.

MH01 and MH03 results are similar illustrating good robustness to movement. The almost undisturbed RPE and APE for MH05, as compared to MH01 and MH03, seem to indicate robustness to lack of texture. However, this is only due to ORB-SLAM2's ability to relocalize and close loops, detailed in Section 7.4 where ORB-SLAM2 was tested without loop closures. The VO base of ORB-SLAM2 is actually very sensitive to MH05 parts where texture is missing. IRSTV's results show that ORB-SLAM2 can perfectly handle large difficult environments. Despite the lack of loops, it outputs rather precise results with a 1.1% APE RMSE error over the traveled distance. No failure due to the environment or initialization issued was observed.

To conclude, ORB-SLAM2 is found to be consistent and robust in the various tested scenarios (including very large scales) and has very reliable optimization and loop closing frameworks. It performs at least as great as other state-of-the-art methods including Vins-Mono despite using only the visual information of a high-speed visual-inertial dataset. Its visual odometry base is very sensitive to lacking texture, but loop closure and relocalization allow to correct the trajectory. However, let us notice that ORB-SLAM2 has a very restrictive keyframe-culling policy, which means that it also outputs fewer poses. This obviously improves the output compared to what is truly estimated. It also means that the results sometimes give several seconds without any pose output because no new keyframe was captured during this time (or it was later culled out) as seen in Figure 8. This can be avoided by turning down the severity of the keyframe management parameters in the code. Results with disabled loop closure are displayed in Section 7.4.

7.3.2. Vins-Mono Result Analysis. Vins-Mono RPE RMSE is among the best of our benchmark. Vins-Mono APE results are the second best as well (along with DSO and behind ORB-SLAM2). Its APE max/RMSE ratio is of 2 to 3, and its overall consistency (σ, max) is comparable to DSO's. However, we obtained less predictable results in MH01 ($\overline{\sigma\text{RMSE}}$ of 11.5%), showing that high speed can actually be beneficial for Vins-Mono's robustness. The shapes of the CDF error plots (Figure 9) also indicate that there are still a few singular errors that deteriorate the overall results.

The most noticeable result of Vins-Mono is the high accuracy of the real scale estimation. e_{scale} values are 2.7% on MH01, 2.8% on MH03, and 0.9% on MH05. RPE and APE are not really affected in MH03 by the increased movement. MH05 results are great too when compared to MH01 or MH03, which indicates robustness to lack of texture.

Tests on the IRSTV dataset output an APE RMSE of 3.3% of the traveled distance, which is a satisfying result for most

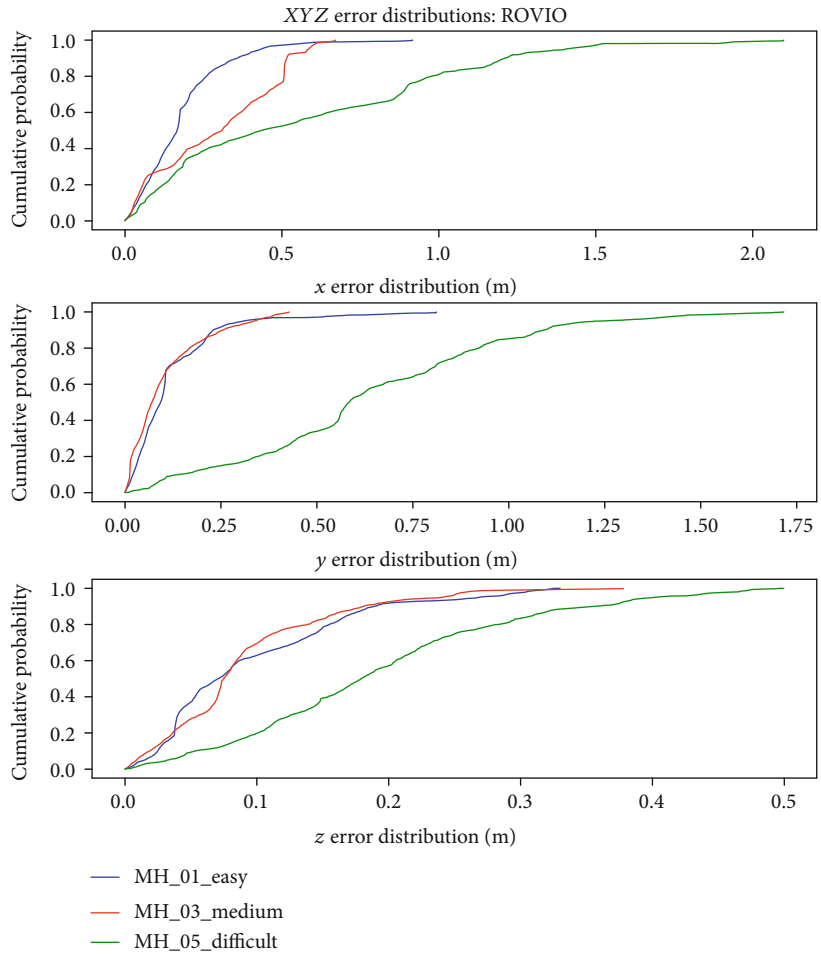


Figure 13: ROVIO: cumulative error distributions alongside each axis of the position on EuRoC MH01, MH03, and MH05.

pedestrian localization applications. However, Vins-Mono struggles a lot with the initialization, probably due to the absence of a specific initialization phase at the beginning of the acquisition, contrary to EuRoC. It can generally initialize when motion with sufficient parallax is detected. However, the quality of the IMU acceleration bias estimated at this moment is uncertain, which greatly impacts the accuracy and robustness of the results. We also found out that glass door reflections were repetitively the cause of failures. Furthermore, the real-scale estimation on the IRSTV dataset is far less precise than in the close environment of EuRoC (around 43% of error!). Nonetheless, we believe these results to be improvable. Adding specific motion for dynamic initialization at the beginning of the dataset would probably have improved the results and robustness.

Globally, Vins-Mono is found to be robust to the difficulties in the presented scenarios, and its great RPE RMSE is promising for lifelong experiments. Although ORB-SLAM2's results are still overall better, the real benefit of Vins-Mono is its ability to accurately estimate the real scale. Consequently, the results can easily be interpreted online as real-world poses on a given map for instance. However, results on the IRSTV dataset show that it can be difficult to meet the requirements in terms of hardware to get the best out of Vins-Mono or that specific user movement might be needed to correctly initialize the IMU bias.

7.3.3. DSO Result Analysis. Along with Vins-Mono and ORB-SLAM2, the DSO method outputs the best RPE RMSE results of all tested methods. Being a pure VO method, its accuracy only depends on the quality of the odometry. DSO's results are also noticeable for their regularity with APE $\bar{\sigma}$ and $\overline{max}$ errors comparable to Vins-Mono's.

DSO is found to be robust to movement since MH03 results are comparable to those of MH01 and robust to lack of texture with MH05 results comparable to those of MH01. Even though DSO is pure visual odometry, its overall results are the second best of our benchmark and are promising for large-scale environments (low RPE RMSE), although Figure 10 shows that the error scale with the length of the trajectory rather than anything else (cf. Table 4). Compared to the other two best methods of our benchmark, DSO also has the advantage of obtaining this accuracy on its live pose estimation and not only on the trajectory reconstruction.

DSO correctly estimates the path and reconstructs the environment for the IRSTV dataset. The scale estimation fails when the camera moves from a street to a large place with a trajectory estimated with two different scales. We only

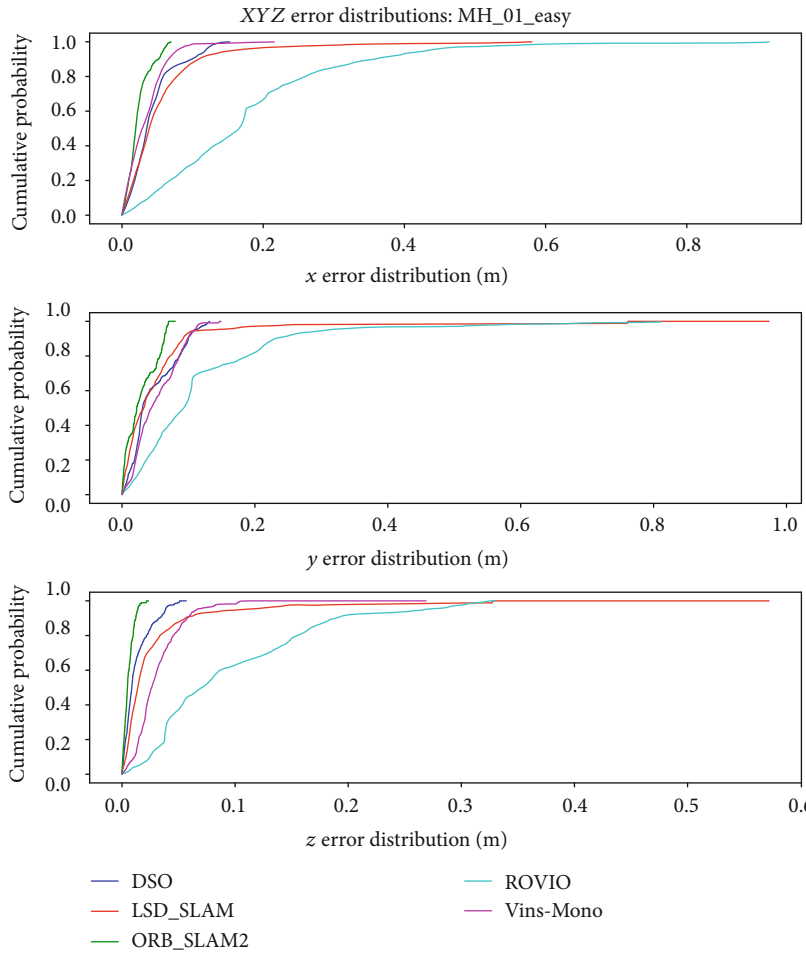


Figure 14: Comparison of position error distributions alongside each axis on EuRoC MH01.

considered the second part of the trajectory for the APE error estimation since it is the only part well reconstructed by the three methods (Figure 11). With a 3.8% APE RMSE over the traveled distance, the results are less precise results than those of ORB-SLAM2 and Vins-Mono. Let us notice that DSO did not face difficulty to initialize, contrary to Vins-Mono cf. Section 7.3.2. The lower accuracy, as well as the scale estimation issue, can be explained by the use of a rolling shutter camera, whereas direct methods perform better with global shutters.

Overall, DSO is precise, consistent, and robust in various environments. The reconstructed semidense map gives an easily readable representation of the observed surroundings. Seeing how great it adapts to various scenarios, the only real weakness of DSO in terms of results is its pure VO nature. It performs great even at very large scale and with difficult environments, but we came across a bad scale-estimation issue. It could be interesting to couple it with a loop closure framework to turn it into a true SLAM method. Also note that DSO requires GPU acceleration, which means that it is not easily adaptable to a wide range of applications.

7.3.4. LSD-SLAM Result Analysis. As far as we know, there are no public tests of LSD-SLAM on EuRoC available. Its package is the oldest of all the algorithms tested here, so we are not refuting the possibility that an adaptation to EuRoC is possible (for instance the Stereo version [90] might be better performing here. (The open-source github project neither seems to be maintained anymore, nor to be upgraded to recent Ubuntu and ROS versions.) However, by using the regular package, parameters, and recommendations accessible today, we managed to run LSD-SLAM on MH03 and MH05. Despite encouraging results on MH01 (Figure 12), its robustness to movement and maybe lack of texture was not sufficient to produce decent results for MH03 and MH05. Also, the tracking failed before the end for every MH03 run we made and on three out of five runs on MH05. Note that to better support the initialization of LSD-SLAM, we cut off the beginning of the dataset, where the drone moves erratically, to initialize its IMU parameters.

Observing the very random behavior obtained on the multiple runs, it seems clear that LSD-SLAM's failure here is due to bad initialization. Most probably, the LSD-SLAM results do not reflect the real abilities of this method in ideal conditions. This result highlights the fact that methods requiring specific attention during initialization are less suited for many applications where such care cannot be brought. The need for specific user intervention and control are critical elements to consider when choosing a SLAM method. With the current settings and initialization process,

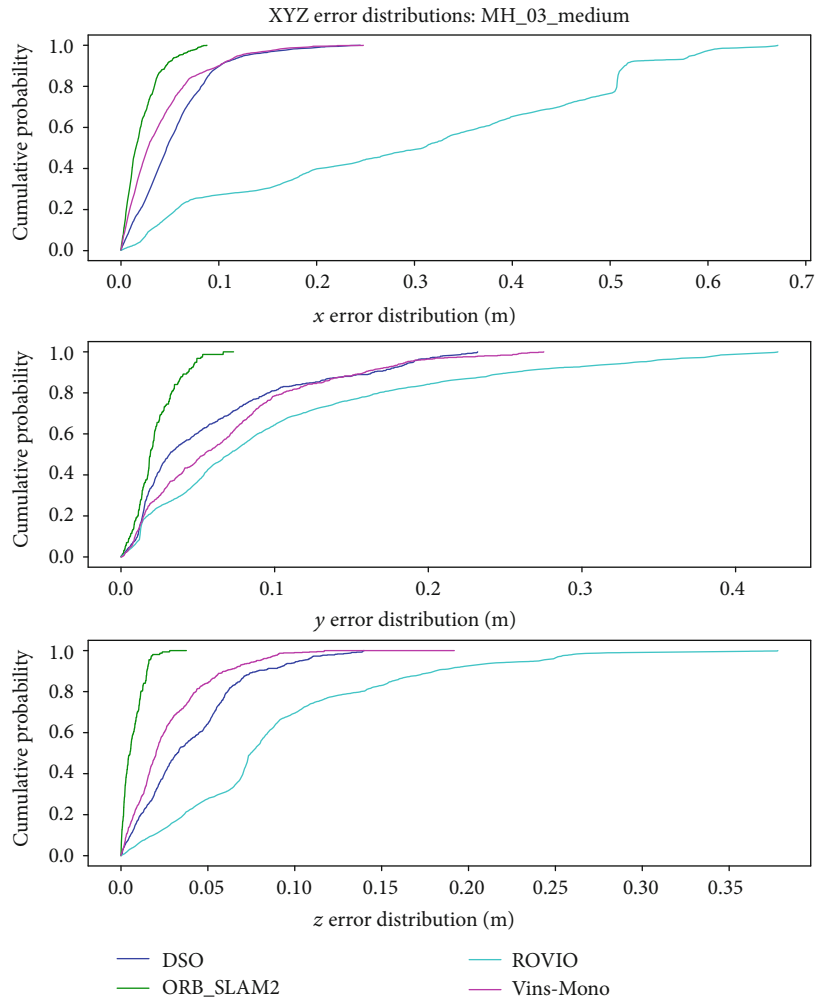


Figure 15: Comparison of position error distributions alongside each axis on EuRoC MH03.

LSD-SLAM was found to be not robust enough for scenarios similar to those presented in EuRoC Machine Hall.

7.3.5. ROVIO Result Analysis. Looking at the dataset MH01 and MH03, among all methods, ROVIO outputs the worst RPE RMSE. The cumulative error distribution plots, shown in Figure 13), show that 80% of the positioning errors for MH01 and MH03 are below 25 cm along the x axis, 20 cm along the y axis, and 15 cm along the z axis. Globally, the approach is still accurate with, for example, the maximal errors along the y axis for MH01 and MH03, with 40 cm and 80 cm, respectively. Overall, it gives one of the worst APE (LSD-SLAM failures apart), with an APE RMSE 3 to 20 times worse than the other methods, and an APE max 3 to 12 times worse. All ROVIO runs give the same results on a dataset; there are no stochastic parameters.

The general scale estimation is good enough for real-scale trajectory estimation. e_{scale} values equal 4.9% for MH01, 4.7% for MH03, and 12.8% for MH05. It seems robust to movement since the performance is not especially altered in MH03 as compared to MH01 (see the APE and RPE). However, the very bad results on MH05 show that it is surprisingly (since it is VIO) badly affected by the passages in textureless environments. Regarding large scales, the lack of precision combined with the pure VIO nature may lead to very imprecise results. However, pure VO, like ROVIO, displays an advantage regarding memory management for lifelong experiments, since it works with a window of keyframes only.

To conclude on ROVIO, it is found to be less precise than other methods. It is only a pure VIO without loop closure and global optimization. It remains an interesting VIO method for online control of drones, for example, or any application that focuses more on local pose estimation than the correctness of the reconstructed trajectory with respect to a reference ground truth. Its predictability is interesting for repetitive tasks and known scenarios. However, textureless environments seem to introduce serious issues and the lack of consistency in the results is problematic for applications that require to continuously assess the precision. ROVIO failed on the IRSTV dataset.

7.4. Comparative Analysis and Conclusion regarding Pedestrian Urban Navigation with Handheld Sensors. Comparative pose estimation results for all the tested methods

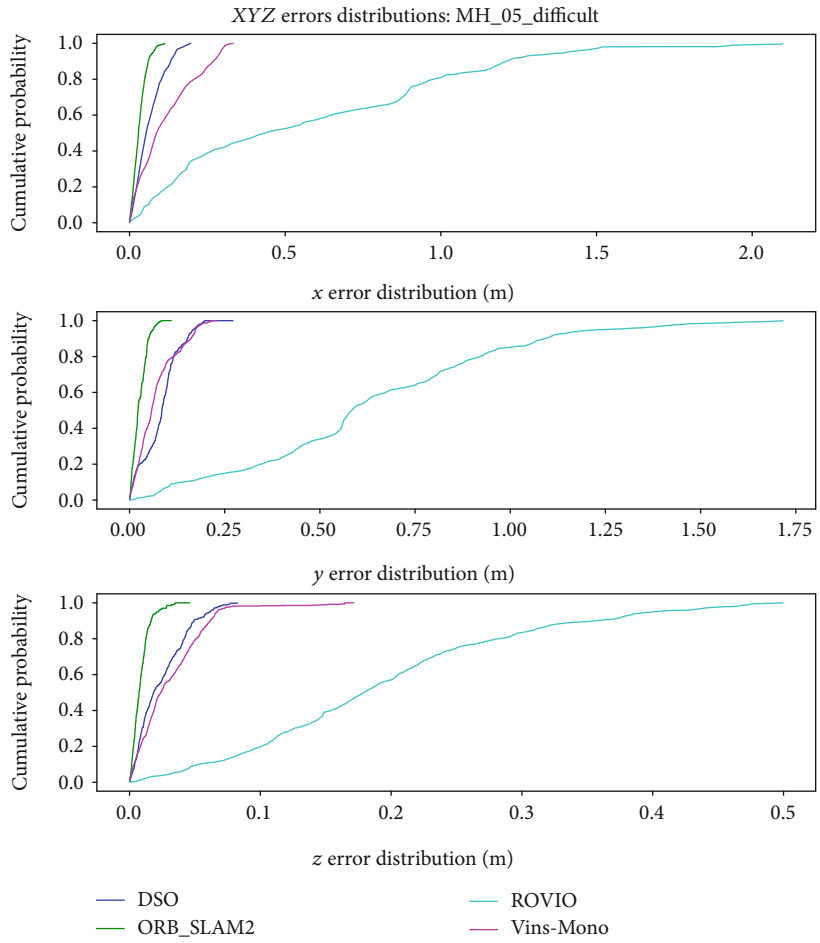


Figure 16: Comparison of position error distributions alongside each axis on EuRoC MH05.

are shown in Figures 14–16. As for rotation estimation only, all methods perform relatively good without any distinctive difference. We observe a small tendency to drift in ROVIO's yaw estimation. It is illustrated with the rotation estimation for MH03 in Figure 17.

Due to the potential lack of loop and our focus on the correctness of live pose estimation, a strong VO or VIO base is needed. Therefore, we added the results for Vins-Mono [74] and ORB-SLAM2 [76] without loop closure in Table 7. A real-scale estimation capability is also needed, as it would greatly ease the development of solutions for online applications.

Table 7 shows that ORB-SLAM2 rarely uses loop closure on the EuRoC tested since the results on MH01 and MH03 are almost the same. However, it uses it at a larger scale to correct drift. In MH05, the passage in the dark introduces a large positioning uncertainty, which means that the trajectory is drifting from there until the end. If no loop closure is possible, the result completely depends on how bad the pose estimation was during the textureless part. Here, the APE RMSE results range from 14 cm up to 3.7 m. Vins-Mono seems to use loop closure more often for the same since its absence doubles the errors. However, thanks to the IMU integration, the error is bounded and more predictable. Choosing ORB-SLAM2 for its more precise results may be a dangerous bet if loop closure is difficult to perform in the considered environment. The reason is that its results are heavily impacted by a problem in pose estimation due to a momentary lack of texture.

Finally, the computation on the IRSTV dataset finds ORB-SLAM2 to be the most robust approach. It handles urban space difficulties (glass reflections, scale changes, and pedestrian motion) and even indoor-to-outdoor transitions since illumination changes do not cause major ORB-SLAM2 failure as in MH05 without loop closure. In conclusion, DSO, Vins-Mono, and ORB-SLAM2 are all suitable choices for our use case, i.e., pedestrian urban navigation with handheld sensors. The choice also depends on the type of hardware available: GPU+global shutter for DSO, tightly synchronized IMU, and camera for Vins-Mono. With the high-end hardware, DSO might be preferred for urban environments with severely textureless places, while Vins-Mono can offer more realistic scale estimation without further manipulation (when properly initialized). However, when considering user-friendliness (i.e., easy initialization), easiness to set up, hardware and computational power requirements, and global robustness and accuracy, ORB-SLAM2 comes at a first choice for our use case.

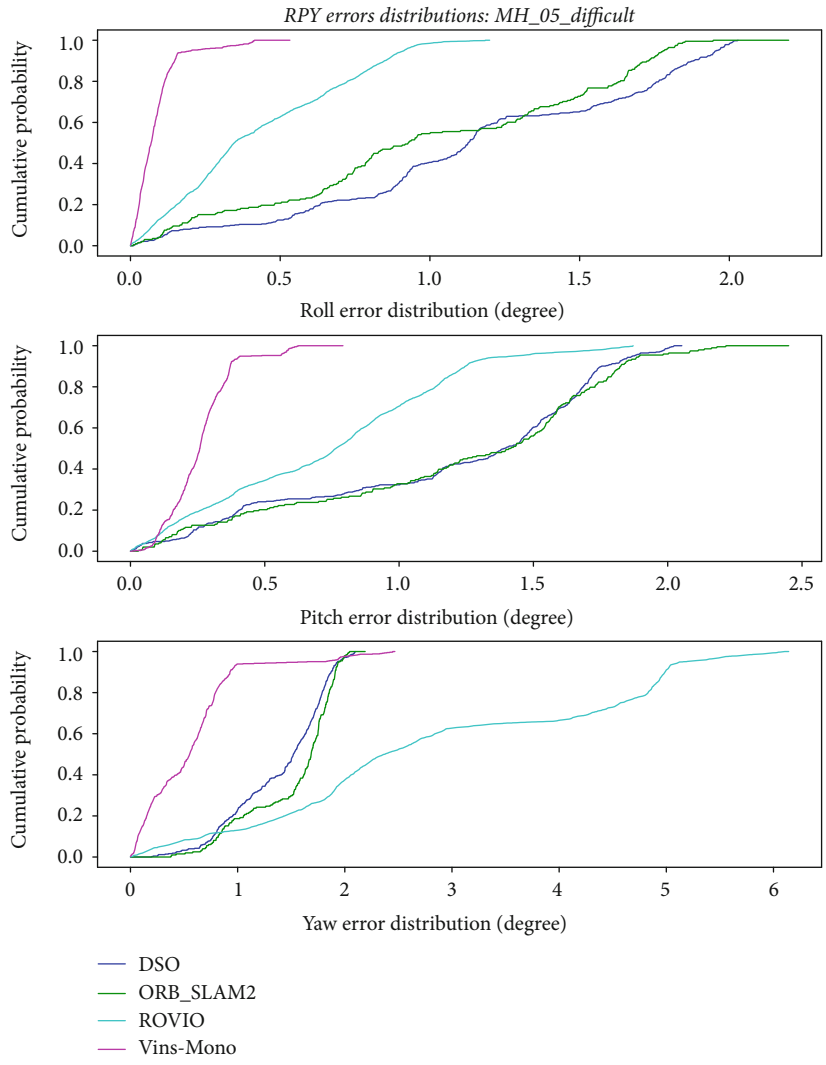


Figure 17: Comparison of rotation error distributions alongside each axis on EuRoC MH05.

Table 7: Vins-Mono and ORB-SLAM2: APE RMSE results on EuRoC with loop closure disabled. Results with loop closure enabled are recalled in brackets [].

APE RMSE (cm)	Vins-Mono no loop	ORB-SLAM2 no loop
Machine Hall 01	15.3 (0.4%) [8.46]	4.62 (1.4%) [4.38]
Machine Hall 03	19.4 (0.1%) [9.51]	3.90 (0.7%) [3.89]
Machine Hall 05	29.3 (0.1%) [17.39]	165 (54%) [5.31]

8. Conclusion

We conducted a review of important SLAM approaches and detailed the core notions of vSLAM and viSLAM along with the different existing designs. We linked this theoretical survey with a historical overview to identify the main milestones in SLAM evolution divided into three main periods. Finally, we classified some of the most famous methods comparing their main design characteristics, their objectives, and their expected robustness in various scenarios using five key features describing the nature of common use cases. Our experimental benchmark focuses on pedestrian pose estimation with a handheld device in urban environments. It emphasizes three reliable SLAM methods: Vins-Mono, DSO, and ORB-SLAM2. Overall, ORB-SLAM2 provides the best performance. However, for applications where real-scale estimation is needed online, an additional framework is required. Such a framework could tackle the lack of loop on very large trajectories that are frequent in pedestrian applications. For example, correcting the pose using known recognized urban locations such as bike stations or bus stops is an interesting solution [91]. It seems interesting to extend the experimental benchmark of vSLAM to test the robustness of existing methods to the five key features used to describe common use cases. It would support assessing these key features specifically (e.g., manually introduce illumination changes in the frames) but also enlarge the assessment on other specific use cases for general and detailed analysis. The work in [3] is one example of it that unfortunately tests only viSLAM methods.

Our use case is located in a dynamic environment. Consequently, it is interesting to use the new semantic SLAM algorithms to differentiate fixed and mobile elements and to assist the process with the environmental features, such as planes [61]. Two other types of information can be added to future work. First, approaches merging maps after several passages in the same area suggest using preexisting maps of the urban space. Indeed, 3D maps are increasingly enriched and distributed, although their update rate remains problematic. Second, we intend to concentrate on better modeling the pattern of individual walking gait to support pedestrian applications and precise urban positioning.

Data Availability

The data used in this article are from the EuRoC dataset from [8] available at https://projects.asl.ethz.ch/datasets/doku.php?id=kmavvisualinertialdatasets. The IRSTV dataset is provided as supplementary material following the same formatting as EuRoC dataset. Available data are as follows: visual-inertial sensor unit–images (VIRB 30 Ultra, 20 Hz)–MEMS IMU (VN300 (VectorNav), 200 Hz)–ground truth STIM300 (Sensonor) in 2D and calibration with (i) camera intrinsics and (ii) camera-IMU extrinsic. They can be downloaded here: 10.5281/zenodo.4415641.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was funded by IRSTV, IFSTTAR, and ECN.

Supplementary Materials

The IRSTV dataset is provided as supplementary material. *(Supplementary Materials)*

References

[1] J. J. Leonard and H. F. Durrant-Whyte, "Simultaneous map building and localization for an autonomous mobile robot," in *Proceedings IROS '91:IEEE/RSJ International Workshop on Intelligent Robots and Systems '91*, pp. 1442–1447, Osaka, Japan, November 1991.

[2] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *The International Journal of Robotics Research*, vol. 5, no. 4, pp. 56–68, 2016.

[3] J. Delmerico and D. Scaramuzza, "A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots," in *2018 IEEE International Conference on Robotics and Automation (ICRA),*, pp. 2502–2509, Brisbane, QLD, Australia, May 2018.

[4] P. Li, T. Qin, B. Hu, F. Zhu, and S. Shen, "Monocular visual-inertial state estimation for mobile augmented reality," in *2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 11–21, Nantes, France, October 2017.

[5] T. Schöps, J. Engel, and D. Cremers, "Semi-dense visual odometry for AR on a smartphone," in *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 145–150, Munich, Germany, September 2014.

[6] M. Burri, J. Nikolic, P. Gohl et al., "The EuRoC micro aerial vehicle datasets," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, 2016.

[7] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rend'on-Mancha, "Visual simultaneous localization and mapping: a survey," *Artificial Intelligence Review*, vol. 43, no. 1, pp. 55–81, 2012.

[8] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual SLAM algorithms: a survey from 2010 to 2016," *IPSJ Transactions on Computer Vision and Applications*, vol. 9, pp. 1–11, 2017.

[9] K. Yousif, A. Bab-Hadiashar, and R. Hoseinnezhad, "An overview to visual odometry and visual SLAM: applications to mobile robotics," *Intelligent Industrial Systems*, vol. 1, no. 4, pp. 289–311, 2015.

[10] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE Robotics Automation Magazine*, vol. 18, no. 4, pp. 80–92, 2011.

[11] F. Fraundorfer and D. Scaramuzza, "Visual odometry: part II: matching, robustness, optimization, and applications," *IEEE Robotics Automation Magazine*, vol. 19, no. 2, pp. 78–90, 2012.

[12] 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.

[13] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *IEEE Robotics Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[14] C. Cadena, L. Carlone, H. Carrillo et al., "Past, present, and future of simultaneous localization and mapping: toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[15] C. Chen, H. Zhu, M. Li, and S. You, "A review of visual-inertial simultaneous localization and mapping from filtering-based and optimization-based perspectives," *Robotics*, vol. 7, no. 3, pp. 1–20, 2018.

[16] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 573–580, Vilamoura, Portugal, October 2012.

[17] K. Sun, K. Mohta, B. Pfrommer et al., "Robust stereo visual inertial odometry for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 965–972, 2018.

[18] M. Smith, I. Baldwin, W. Churchill, R. Paul, and P. Newman, "The new college vision and laser data set," *The International Journal of Robotics Research*, vol. 28, no. 5, pp. 595–599, 2009.

[19] J. Engel, V. Usenko, and D. Cremers, "A photometrically calibrated benchmark for monocular visual odometry," 2016, https://arxiv.org/abs/1607.02555.

[20] K. Kim, M. Billinghurst, G. Bruder, H. B. L. Duh, and G. F. Welch, "Revisiting trends in augmented reality research: a review of the 2nd decade of ISMAR (20082017)," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 11, pp. 2947–2962, 2018.

[21] 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.

[22] E. Eade and T. Drummond, "Scalable monocular SLAM," in *2006 IEEE Computer Society Conference on Computer Vision*

and Pattern Recognition - Volume 1 (CVPR'06), vol. 1, pp. 469–476, New York, NY, USA, USA, June 2006.

[23] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: a factored solution to the simultaneous localization and mapping problem," in *In Proceedings of the AAAI National Conference on Artificial Intelligence*, pp. 593–598, 2002.

[24] M. Li and A. I. Mourikis, "High-precision, consistent EKF-based visual-inertial odometry," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, 2013.

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

[26] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct EKF-based approach," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 298–304, Hamburg, Germany, September 2015.

[27] 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, Nara, Japan, November 2007.

[28] C. Harris and M. Stephens, "A combined corner and edge detection," in *Proceedings of The Fourth Alvey Vision Conference*, pp. 147–151, University of Manchester, 1988.

[29] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: speeded up robust featuresSpringer," in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds., pp. 404–417, Berlin, Heidelberg, 2006.

[30] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[31] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision – ECCV 2006. ECCV 2006. Lecture Notes in Computer Science, vol 3951*, A. Leonardis, H. Bischof, and A. Pinz, Eds., pp. 430–443, Springer, Berlin, Heidelberg, 2006.

[32] 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, Barcelona, Spain, November 2011.

[33] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," 2016, https://arxiv.org/abs/1607.02565.

[34] R. A. Newcombe, S. Lovegrove, and A. J. Davison, "DTAM: dense tracking and mapping in real-time," in *2011 International Conference on Computer Vision*, pp. 2320–2327, Barcelona, Spain, November 2011.

[35] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: large-scale direct monocular SLAM," in *Computer Vision – ECCV 2014. ECCV 2014*, vol. 8690 of Lecture Notes in Computer Science, Springer, Cham.

[36] P. F. Alcantarilla, J. J. Yebes, J. Almaz'an, and L. M. Bergasa, "On combining visual SLAM and dense scene flow to increase the robustness of localization and mapping in dynamic environments," in *2012 IEEE International Conference on Robotics and Automation*, pp. 1290–1297, Saint Paul, MN, USA, May 2012.

[37] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[38] R. Smith, M. Self, and P. Cheeseman, "A stochastic map for uncertain spatial relationships," in *Proceedings of the 4th International Symposium on Robotics Research*, pp. 467–474, Cambridge, MA, USA, 1988.

[39] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[40] Y. Chen and G. Medioni, "Object modeling by registration of multiple range images," in *Proceedings 1991 IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2724–2729, Sacramento, CA, USA, April 1991.

[41] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, pp. 145–152, Quebec City, Quebec, Canada, May-June 2001.

[42] J. Sola, "Consistency of the monocular EKF-SLAM algorithm for three different landmark parametrizations," in *2010 IEEE International Conference on Robotics and Automation*, pp. 3513–3518, Anchorage, AK, USA, May 2010.

[43] H. Huang, Y. Sun, H. Ye, and M. Liu, "Metric monocular localization using signed distance fields," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1195–1201, Macau, China, November 2019.

[44] B. Triggs, P. Mclauchlan, R. Hartley et al., "Bundle adjustment, a modern synthesis," in *Vision Algorithms: Theory and Practice. IWVA 1999. Lecture Notes in Computer Science, vol 1883*, B. Triggs, A. Zisserman, and R. Szeliski, Eds., Springer, Berlin, Heidelberg, 2010.

[45] S. Agarwal K. Mierle et al., "Ceres solver," http://ceres-solver.org.

[46] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G^2o: a general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*, pp. 3607–3613, Shanghai, China, May 2011.

[47] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "ISAM2: incremental smoothing and mapping using the Bayes tree," *International Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, 2012.

[48] 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.

[49] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: binary robust independent elementary features," in *Computer Vision – ECCV 2010. ECCV 2010*, K. Daniilidis, P. Maragos, and N. Paragios, Eds., vol. 6314 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2010.

[50] R. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME - Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[51] P. S. Maybeck and G. M. Siouris, "Stochastic models, estimation, and control, volume I," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 10, no. 5, pp. 282–282, 1980.

[52] H. F. Durrant-Whyte, "Uncertain geometry in robotics," *IEEE Journal on Robotics and Automation*, vol. 4, no. 1, pp. 23–31, 1988.

[53] H. Durrant-Whyte, D. Rye, and E. Nebot, "Localization of autonomous guided vehicles," in *Robotics Research*, G. Giralt and G. Hirzinger, Eds., pp. 613–625, Springer, London, 1996.

[54] S. Weiss, M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart, "Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments," in *2012 IEEE*

International Conference on Robotics and Automation, pp. 957–964, Saint Paul, MN, USA, May 2012.

[55] Y. Sun, M. Liu, and M. Q. H. Meng, "Improving RGB-D SLAM in dynamic environments: a motion removal approach," *Robotics and Autonomous Systems*, vol. 89, pp. 110–122, 2017.

[56] Y. Sun, M. Liu, and M. Q. H. Meng, "Motion removal for reliable RGB-D SLAM in dynamic environments," *Robotics and Autonomous Systems*, vol. 108, pp. 115–128, 2018.

[57] J. Cheng, Y. Sun, and M. Q. H. Meng, "Improving monocular visual SLAM in dynamic environments: an optical-flow-based approach," *Advanced Robotics*, vol. 33, no. 12, pp. 576–589, 2019.

[58] L. Xiao, J. Wang, X. Qiu, R. Zheng, and X. Zou, "Dynamic-SLAM: semantic monocular visual localization and mapping based on deep learning in dynamic environment," *Robotics and Autonomous Systems*, vol. 117, pp. 1–16, 2019.

[59] M. Henein, J. Zhang, R. Mahony, and V. Ila, "Dynamic SLAM: the need for speed," 2020, http://arxiv.org/abs/2002.08584.

[60] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: an open-source library for real-time metric-semantic localization and mapping," in *2020 IEEE International Conference on Robotics and Automation (ICRA),*, Paris, France, May 2020.

[61] S. Yang, S. Yu, M. Kaess, and S. Scherer, "Pop-up SLAM: semantic monocular plane SLAM for low-texture environments," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1222–1229, Daejeon, South Korea, November 2016.

[62] T. Schneider, M. Dymczyk, M. Fehr et al., "Maplab: an open framework for research in visual-inertial mapping and localization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1418–1425, 2018.

[63] A. R. Vidal, H. Rebecq, T. Horstschaefer, and D. Scaramuzza, "Ultimate SLAM? Combining events, images, and IMU for robust visual SLAM in HDR and high-speed scenarios," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 994–1001, 2018.

[64] H. Liu, M. Chen, G. Zhang, H. Bao, and Y. Bao, "ICE-BA: incremental, consistent and efficient bundle adjustment for visual-inertial SLAM," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1974–1982, Salt Lake City, UT, USA, June 2018.

[65] G. Klein and D. W. Murray, "Improving the agility of keyframe-based SLAM," in *Computer Vision – ECCV 2008. ECCV 2008. Lecture Notes in Computer Science, vol 5303*, D. Forsyth, P. Torr, and A. Zisserman, Eds., Springer, Berlin, Heidelberg, 2008.

[66] P. Ondrúška, P. Kohli, and S. Izadi, "Mobilefusion: real-time volumetric surface reconstruction and dense tracking on mobile phones," *IEEE Transactions on Visualization and Computer Graphics*, vol. 21, no. 11, pp. 1251–1258, 2015.

[67] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "SVO: semidirect visual odometry for monocular and multicamera systems," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 249–265, 2017.

[68] R. A. Newcombe, S. Izadi, O. Hilliges et al., "Kinectfusion: real-time dense surface mapping and tracking," in *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pp. 127–136, Basel, Switzerland, October 2011.

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

[70] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison, *Elasticfusion: Dense SLAM without a Pose Graph*, Robotics: Science and Systems, 2015.

[71] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Multi-sensor fusion for robust autonomous flight in indoor and outdoor environments with a rotorcraft MAV," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4974–4981, Hong Kong, China, May 2014.

[72] I. Cvišić, J. Ćesić, I. Marković, and I. Petrović, "SOFT-SLAM: computationally efficient stereo visual simultaneous localization and mapping for autonomous unmanned aerial vehicles," *Journal of Field Robotics*, vol. 35, 2017.

[73] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[74] T. Qin, P. Li, and S. Shen, "VINS-mono: a robust and versatile monocular visualinertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2017.

[75] R. Mur-Artal and J. D. Tardos, "Visual-inertial monocular SLAM with map reuse," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 796–803, 2017.

[76] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2016.

[77] C. Chen, H. Zhu, L. Wang, and Y. Liu, "A stereo visual-inertial SLAM approach for indoor mobile robots in unknown environments without occlusions," *IEEE Access*, vol. 7, pp. 185408–185421, 2019.

[78] B. Clipp, J. Lim, J. M. Frahm, and M. Pollefeys, "Parallel, real-time visual SLAM," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3961–3968, Taipei, Taiwan, October 2010.

[79] H. Strasdat, A. J. Davison, J. M. M. Montiel, and K. Konolige, "Double window optimisation for constant time visual SLAM," in *2011 International Conference on Computer Vision*, pp. 2352–2359, Barcelona, Spain, November 2011.

[80] K. Pirker, M. Rüther, and H. Bischof, "CD SLAM - continuous localization and mapping in a dynamic world," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3990–3997, San Francisco, CA, USA, September 2011.

[81] S. Maity, A. Saha, and B. Bhowmick, "Edge SLAM: edge points based monocular visual SLAM," in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pp. 2408–2417, Venice, Italy, October 2017.

[82] T. Whelan, M. Kaess, M. F. Fallon, H. Johannsson, J. J. Leonard, and J. McDonald, "Kintinuous: spatially extended kinectfusion," CSAIL Technical Reports, 2012.

[83] B. Pfrommer, N. Sanket, K. Daniilidis, and J. Cleveland, "PennCOSYVIO: a challenging visual inertial odometry benchmark," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3847–3854, Singapore, Singapore, May-June 2017.

[84] A. L. Majdik, C. Till, and D. Scaramuzza, "The Zurich urban micro aerial vehicle dataset," *The International Journal of Robotics Research*, vol. 36, no. 3, pp. 269–273, 2017.

[85] D. Schubert, T. Goll, N. Demmel, V. Usenko, J. Stuckler, and D. Cremers, "The TUM VI benchmark for evaluating visual-inertial odometry," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1680–1687, Madrid, Spain, October 2018.

[86] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Providence, RI, USA, June 2012.

[87] M. Ortiz, M. De Sousa, and V. Renaudin, "A new PDR navigation device for challenging urban environments," *Journal of Sensors*, vol. 2017, 11 pages, 2017.

[88] J. Le Scornec, M. Ortiz, and V. Renaudin, "Foot-mounted pedestrian navigation reference with tightly coupled GNSS carrier phases, inertial and magnetic data," in *2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Sapporo, Japan, September 2017.

[89] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 376–380, 1991.

[90] J. Engel, J. Stuckler, and D. Cremers, "Large-scale direct SLAM with stereo cameras," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1935–1942, Hamburg, Germany, September 2015.

[91] N. Antigny, M. Servieres, and V. Renaudin, "Fusion of 3D GIS, vision, inertial and magnetic data for improved urban pedestrian navigation and augmented reality applications," *Navigation: Journal of The Institute of Navigation*, vol. 65, no. 3, pp. 431–447, 2018.