Keyframe-based monocular SLAM: design, survey, and future directions

CrossMark

Georges Younes [a,b], Daniel Asmar [a,*], Elie Shammas [a], John Zelek [b]

[a] *Vision and Robotics Lab, Mechanical Engineering Department, American University of Beirut, Beirut, Lebanon*
[b] *Systems Design Department, University of Waterloo, Waterloo, Ont., Canada*

HIGHLIGHTS

- A general guideline for the design of a Keyframe-based monocular SLAM.
- A survey on all Keyframe-based monocular SLAM (KSLAM) systems
- Insight into the current state of KSLAM and future directions

ARTICLE INFO

ABSTRACT

Extensive research in the field of monocular SLAM for the past fifteen years has yielded workable systems that found their way into various applications in robotics and augmented reality. Although filter-based monocular SLAM systems were common at some time, the more efficient keyframe-based solutions are becoming the de facto methodology for building a monocular SLAM system. The objective of this paper is threefold: first, the paper serves as a guideline for people seeking to design their own monocular SLAM according to specific environmental constraints. Second, it presents a survey that covers the various keyframe-based monocular SLAM systems in the literature, detailing the components of their implementation, and critically assessing the specific strategies made in each proposed solution. Third, the paper provides insight into the direction of future research in this field, to address the major limitations still facing monocular SLAM; namely, in the issues of illumination changes, initialization, highly dynamic motion, poorly textured scenes, repetitive textures, map maintenance, and failure recovery.

© 2017 Elsevier B.V. All rights reserved.

1. Introduction

When we enter a room or a space, we assess our surroundings, build a map, and keep track of our location with respect to all the objects in the space. Simultaneous Localization and Mapping (SLAM) is the equivalent of this procedure for a machine. A robot's location in the map is needed for various tasks such as navigation, surveillance and manipulation, to name a few. SLAM can be performed with various sensors and their combinations, such as cameras, LIDARS, range finders, GPS, IMU, etc. With more information such as depth, location, and velocity are available, the easier the problem. If only a single camera is present, the problem is more challenging since they are bearing only sensors; however, the rewards are great because a single camera is passive, consumes low power, is of low weight, needs a small physical space, is inexpensive, and ubiquitously found in hand-held devices. Furthermore, cameras can operate across different types of environments,

both indoor and outdoor, in contrast to active sensors such as infrared based RGB-D sensors that are sensitive to sunlight. For the aforementioned reasons, this paper handles SLAM solutions using a single camera only and is referred to as monocular SLAM. Even though there are still many challenges facing monocular SLAM, the research community is actively seeking and finding solutions to these problems.

A number of surveys for the general SLAM problem exist in the literature, but only a few of them handle monocular SLAM in an exclusive manner. The most recent survey on SLAM is the one by [1], which discusses the complete SLAM problem, but does not delve into the specifics of keyframe-based monocular SLAM, as we do in this paper. In 2011, [2] published a tutorial on Visual odometry, but did not detail the solutions put forward by the research community; rather, it presented a generic design for earlier visual odometry systems. In 2012, [3] published a general survey on Visual SLAM, but also did not detail the solutions put forward by different people in the community. Also, subsequent to the date of the aforementioned papers were published, almost thirteen new systems have been proposed, with many of them introducing

* Corresponding author.
E-mail address: da20@aub.edu.lb (D. Asmar).

significant contributions to keyframe-based Monocular SLAM. In 2015, [4] also published a survey on Visual SLAM with the main focus on filter-based methods, visual odometry (VO), and RGB-D systems. While filter-based Visual SLAM solutions were common before 2010, most solutions thereafter designed their systems around a *non-filter*, keyframe-based architecture. The survey of Yousif et al. describes a generic Visual SLAM but lacks focus on the details and problems of monocular keyframe-based systems.

To our knowledge, this paper is unique in that it systematically discusses the different components of keyframe-based monocular SLAM systems, while highlighting the nuances that exist in their different implementations. Our paper is a valuable tool for any scholar or practitioner in the field of localization and mapping. With the many new proposed systems appearing every day, the information is daunting for the novice and people are often perplexed as to which algorithm they should use. Furthermore, this paper produces structure for researchers to quickly pinpoint the shortcomings of each of the proposed techniques and accordingly help them focus their effort on alleviating these weaknesses.

The remainder of the paper is structured as follows. Section 2 presents the architecture of a generic keyframe-based Monocular SLAM, and details the particulars of its major building blocks, mainly: data association, visual initialization, pose estimation, topological/metric map generation, Bundle Adjustment (BA)/Pose Graph Optimization (PGO)/map maintenance, and global localization (failure recovery and loop closure). Sections 3 and 4 respectively survey all the open-source and closed-source keyframe-based Monocular SLAM systems in the literature. Open-source systems are treated in more depth, given the additional insight gained through access to their code. In Section 5, we delve into the traits of the seven building blocks and explain how to best implement them for different environmental conditions. Finally, we provide in the conclusion our insight into the current open problems in monocular SLAM and we propose possible venues for the solution of each of these problems.

2. Keyframe-based monocular SLAM architecture

Monocular SLAM solutions are either filter-based, such as using a Kalman filter; or keyframe-based, relying on optimization to estimate both motion and structure. In filter-based systems, the localization and mapping are intertwined: the camera pose T_n, with the entire state of all landmarks in the map, are tightly joined and need to be updated at every processed frame as shown in Fig. 1-a. On the other hand, in keyframe-based systems, *Localization* and *Mapping* are separated into two steps: camera localization takes place on regular frames over a subset of the map (gray nodes in Fig. 1-b), whereas keyframe-based optimization takes place on *Keyframes*. As a consequence of these differences, Strasdat et al. in 2010 showed that keyframe based methods outperform filter-based ones; and it is therefore not surprising to note that most new releases of monocular SLAM systems are keyframe-based.

For this reason and the fact that filter-based SLAM have been relatively well covered in the literature in many surveys ([3–6], etc.), the focus of this paper will be on the analysis and survey of only Keyframe-based monocular SLAM systems, hereafter referred to as KSLAM.

2.1. KSLAM architecture

Designing a KSLAM system requires the treatment of seven main components (Fig. 2); namely (1) visual initialization, (2) data association, (3) pose estimation, (4) topological/metric map generation, (5) BA/PGO/map maintenance, (6) failure recovery and (7) loop closure. Fig. 2 describes the general flow between these modules: at startup, KSLAM has no prior information about the

camera pose nor the scene structure, the visual initialization module is responsible for establishing an initial 3D map and the first camera poses in the system. The visual initialization is the entry point of a KSLAM and runs only once at startup. When a new frame is available, data association uses the previous camera poses to guess a pose for the new frame; the predicted pose is used to establish associations with the 3D map. An error vector is then found as the difference between the *true* measurements and their associated *matches* generated using the guessed pose. The error vector is iteratively minimized in the pose optimization module, using the guessed pose as a starting point. If the minimization diverges or the data association fails, failure recovery procedures are invoked. For regular frames, the pipeline ends here however, if the frame was chosen as a keyframe, it is used to triangulate new landmarks, thus expanding the 3D map. To ensure the coherency of the map, reduce errors and remove outliers, map maintenance continuously optimizes the map while another parallel process attempts to detect loop closures and accordingly minimize the errors accumulated over the traversed loop.

Fig. 3 presents the different solutions for each of the KSLAM modules; we further elaborate on these solutions in the upcoming sections.

2.2. Data association

VSLAM methods are categorized based on the type of information they process from the images, as being direct, feature-based, or a hybrid of both.

2.2.1. Design choice
a. Direct methods. Direct methods are categorized into either dense or semi-dense methods: dense methods exploit the information available at every pixel, whereas semi-dense methods exploit the information from every pixels at which the gradient of image brightness is significant. Fig. 3-A shows an example of a direct approach: pixel values surrounding a location of interest (in this case a triangle) are aligned by a *transformation* that minimizes the intensity values between the two locations of interest in both images. In practice a region of interest is defined as a square surrounding a pixel.

The basic underlying principle for all direct methods is known as the brightness consistency constraint and is best described as:

$$J(x, y, t) = I(x + u(x, y), y + v(x, y), t + 1), \tag{1}$$

where x and y are pixel coordinates; u and v denote displacement functions of the pixel (x, y) between two images I and J of the same scene taken at time t and $t + 1$ respectively. The brightness consistency constraint is based on the assumption that a point from the world's surface, observed in image I, will have the same intensity when observed in a subsequent image J. To render Eq. (1) solvable, [7] suggested, in what they referred to as Forward Additive Image Alignment (FAIA), to replace all the individual pixel displacements u and v by a single general motion model, in which the number of parameters is dependent on the implied type of motion. FAIA iteratively minimizes the squared pixel-intensity difference between the two images over the transformation parameters p:

$$\operatorname*{argmin}_p \sum_{x,y} [I(W(x, y, p)) - J(x, y)]^2, \tag{2}$$

where $W(., ., p)$ is a warping transform that encodes the relationship relating the two images and p corresponds to the parameters of the transform. Eq. (2) is non-linear and requires an iterative non-linear optimization process, with a computational complexity of $O(n^2N + n^3)$ per iteration, where n is the number of parameters
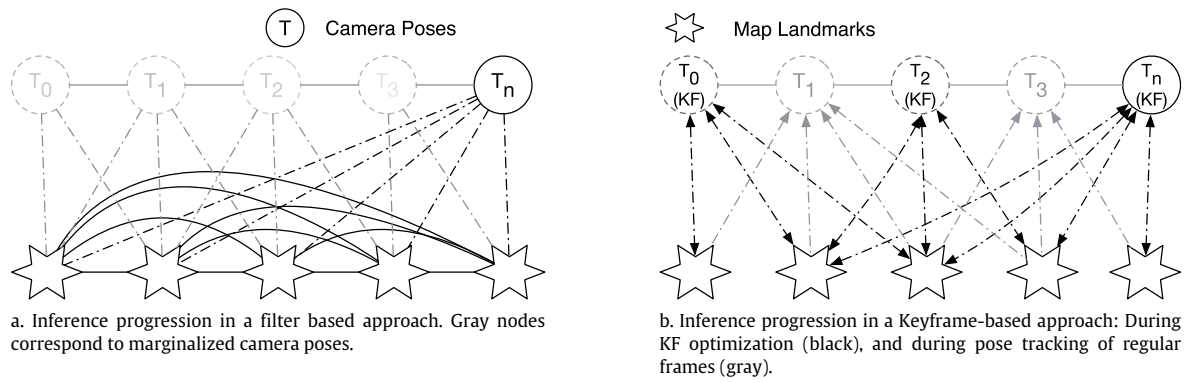
a. Inference progression in a filter based approach. Gray nodes correspond to marginalized camera poses.

b. Inference progression in a Keyframe-based approach: During KF optimization (black), and during pose tracking of regular frames (gray).

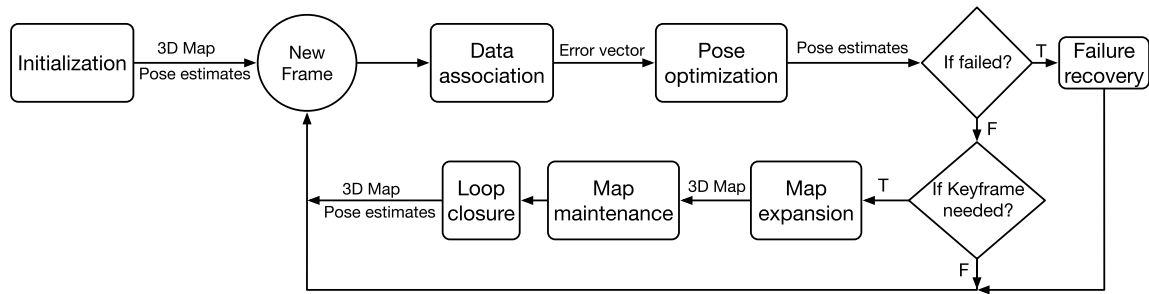**Fig. 1.** Inference in filter based vs. Keyframe-based VSLAM.

Fig. 2. A generic KSLAM flowchart made of 7 components: Visual initialization, data association, pose optimization, topological/metric map generation (map expansion), BA/PGO/map maintenance, failure recovery and loop closure. The basic types of information, returned/manipulated by each component, are overlaid over their corresponding arrows, namely 3D map, pose estimates and error vector.

in p and N is the number of pixels in the image. Since 1981, other variants of the FAIA were suggested such as FCIA (Forward Compositional Image Alignment), ICIA (Inverse Compositional Image Alignment) and IAIA (Inverse Additive Image Alignment) each with different computational complexities. A detailed comparison between these variations can be found in [8].

b. Feature-based methods. Feature-based methods were introduced to reduce the computational complexity of processing each pixel; this is done by matching only salient image locations, referred to as features, or keypoints. An example of feature-based matching is shown in Fig. 3-B. A descriptor is associated to each feature, which is used to provide a quantitative measure of similarity to other keypoints. On one hand, features are expected to be distinctive, invariant to viewpoint and illumination changes, as well as resilient to blur and noise; on the other hand, it is desirable for feature extractors to be computationally efficient and fast. Unfortunately, such objectives are hard to achieve at the same time causing a trade-off between computational speed and feature quality.

The computer vision community has developed, over decades of research, many different feature extractors and descriptors, each exhibiting varying performances in terms of rotation and scale invariance, as well as speed [9]. The selection of an appropriate feature detector depends on the platform's computational power and the type of environment. Feature detector examples include the Hessian corner detector [10], Harris detector [11], Shi–Tomasi corners [12], Laplacian of Gaussian detector [13], MSER [14], Difference of Gaussian [15] and the accelerated segment test family of detectors (FAST, AGAST, OAST) [16].

Feature descriptors include, and are not limited to, BRIEF [17], BRISK [18], SURF [19], SIFT [20], HoG [21], FREAK [22], ORB [23], and a low level local patch of pixels. Further information regarding feature extractors and descriptors is outside the scope of

this work, but the reader can refer to [24], [25,26], or [27] for comparisons.

c. Hybrid methods. Different from the direct and feature-based methods, some systems such as SVO are considered hybrids, which use a combination of both to refine the camera pose estimates, or to generate a dense/semi-dense map. Once a design is chosen, data association is defined as the process of establishing measurement correspondences across different images using either 2D–2D, 3D–2D, or 3D–3D correspondences. The different types of data association are depicted in Fig. 3-C.

2.2.2. Data association types
2D–2D. In 2D–2D correspondence, the 2D feature's location in an image I_2 is sought, given its 2D position in a previously acquired image I_1. Depending on the type of information available, 2D–2D correspondences can be established in one of two ways: when a map is not available and neither the camera transformation between the two frames nor the scene structure is available (*i.e.*, during system initialization), 2D–2D data association is established through a search window surrounding the feature's location from I_1 in I_2. When the transformation relating I_1 and I_2 is known (*i.e.*, the camera pose is successfully estimated), 2D–2D data correspondences are established through epipolar geometry, where a feature in I_1 is mapped to a line in I_2, and the two dimensional search window collapses to a one dimensional search along a line. This latter case often occurs when the system attempts to triangulate 2D features into 3D landmarks during map generation. To limit the computational expenses, a bound is imposed on the search region along the epipolar line.

In both methods, each feature has associated with it a descriptor, which can be used to provide a quantitative measure of similarity to other features. The descriptor similarity measure varies with the type of descriptors used; for example, for a local patch
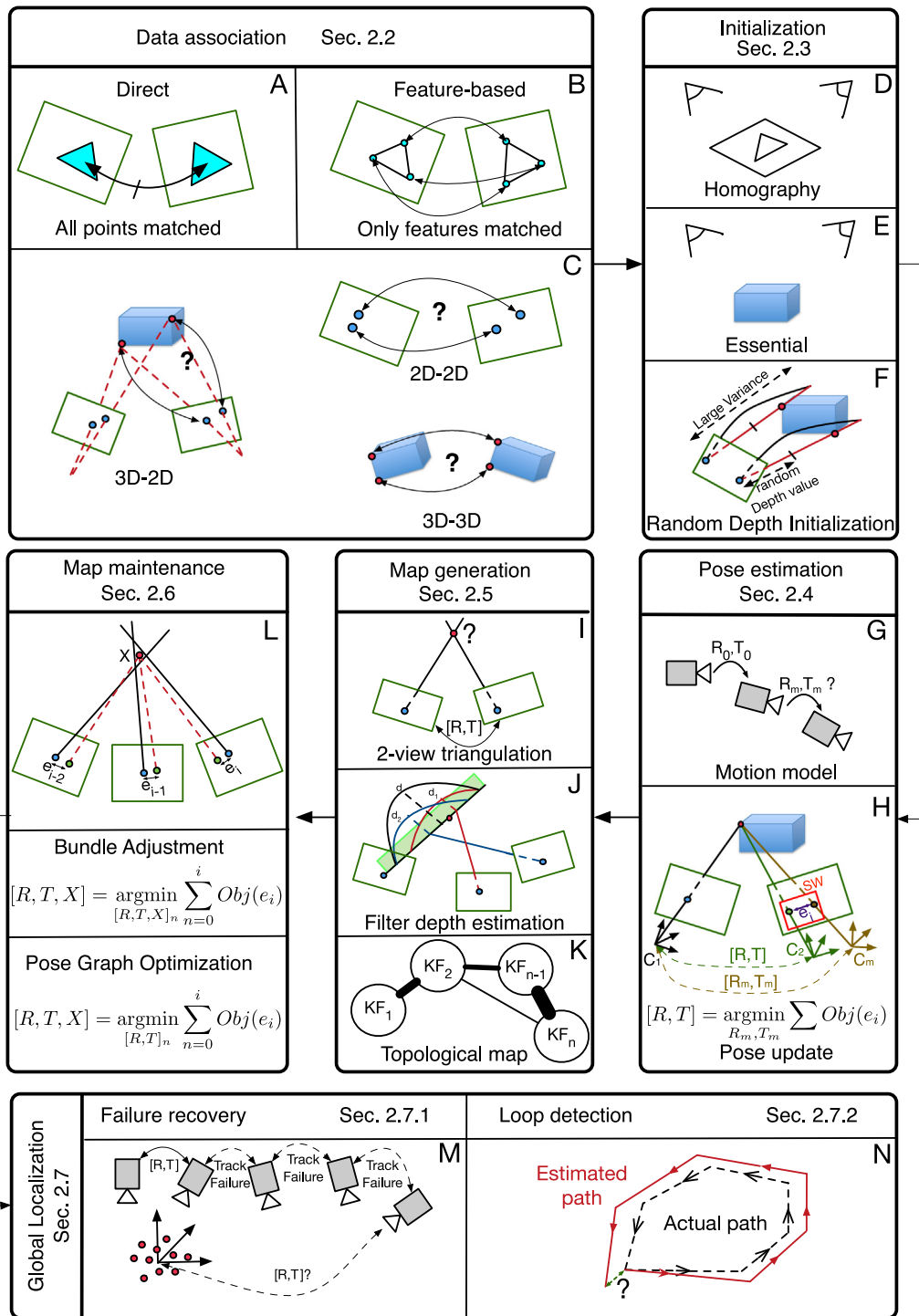
Fig. 3. A graphic representation for the different building blocks of a generic KSLAM.

of pixels, it is typical to use the sum of squared difference (SSD), or a Zero-Mean SSD score (ZMSSD) to increase robustness against illumination changes, as is done in [28]. For higher order feature descriptors – such as ORB, SIFT, or SURF – the L1-norm, the L2-norm, or Hamming distances may be used; however, establishing matches using these measures is computationally intensive and may, if not carefully applied, degrade real-time performance. For such purpose, special implementations that sort and perform feature matching in KD trees, or bags of words, are usually employed. Examples include the works of [29], and [30].

3D–2D. In 3D–2D data association, the camera pose and the 3D structure are known, and one seeks to estimate correspondences between the 3D landmarks and their 2D projection onto a newly acquired frame, without the knowledge of the new camera pose (T, R). This type of data association is typically used during the pose estimation phase of KSLAM. To solve this problem, previous camera poses are exploited in order to yield a hypothesis on the new camera pose and accordingly, project the 3D landmarks onto that frame. 3D–2D data association then proceeds similarly to 2D–2D feature matching, by defining a search window surrounding the

projected location of the 3D landmarks and searching for matching feature descriptors.

3D–3D. 3D–3D data association is typically employed to estimate and correct accumulated drift along loops: when a loop closure is detected, descriptors of 3D landmarks, visible in both ends of the loop, are used to establish matches among landmarks that are then exploited – as explained in [31] – to yield a similarity transform between the frames at both ends of the loop.

2.3. Visual initialization

Monocular cameras are bearing-only sensors, which cannot directly perceive depth; nevertheless, a scaled depth can be estimated via temporal stereoscopy, after observing the same scene through at least two different viewpoints. After KSLAM is initialized, camera pose and 3D structure build on each other, in a heuristic manner, to propagate the system in time, by expanding the map to previously unobserved scenes, while keeping track of the camera pose in the map.

The problem is more difficult during initialization, since neither pose, nor structure is known. In early monocular SLAM systems, such as in MonoSLAM [32], initialization required the camera to be placed at a known distance from a planar scene, composed of four corners of a two dimensional square; the user initialized SLAM by keying in the distance separating the camera from the square. Thereafter, to lessen these constraints, researchers adopted the methods developed by [33] to simultaneously recover the camera pose and the 3D scene structure. Higgins' intuition was to algebraically eliminate the depth from the problem, yielding both the Essential and the Homography matrices. However, the elimination of depth has significant ramifications on the recovered data: since the exact camera motion between the two views cannot be recovered, the camera translation vector is recovered up to an unknown scale λ. Since the translation vector between the two views defines the baseline used to triangulate 3D landmarks, scale loss also propagates to the recovered 3D landmarks, yielding a scene that is also scaled by λ. Fig. 3-D and E describes the two-view initialization using a Homography (which assumes the observed scene to be planar) and an Essential matrix (which assumes the observed scene to be non-planar).

Fig. 4 shows the flowchart of a generic model-based initialization; the first frame processed by the KSLAM system is typically set as the first keyframe. Subsequent frames are processed by establishing 2D–2D data associations, which are monitored to decide whether the new frame is the second keyframe or not. The decision criteria is based on the 2D distances between the found matches in both images. The matches are then used to estimate a Homography (degenerate for non-planar scenes) or a Fundamental matrix (degenerate for planar scenes) using a robust model fitting method (RANSAC or MLESAC [34]). The estimated Homography or the Fundamental matrix are then decomposed as described in [35] into an initial scene structure and initial camera poses. To mitigate degenerate cases, *random depth* initialization (shown in Fig. 3-F), as its name suggests, initializes a KSLAM by randomly assigning depth values with large variance to a single initializing keyframe. The random depth is then iteratively updated over subsequent frames until the depth variance converges.

2.4. Pose estimation

2.4.1. Motion models
Fig. 5 presents a generic flowchart for pose estimation. Because data association is computationally expensive, most monocular SLAM systems assume, for the pose of each new frame, a prior, which guides and limits the amount of work required for data association. Estimating this prior (depicted in Fig. 3-G) is generally

the first task in pose estimation: data association between the two frames is not known yet and one seeks to estimate a prior on the pose of the second frame (T, R), given previously estimated poses.

Most systems employ a *constant velocity* motion model that assumes a smooth camera motion and use the pose changes across the two previously tracked frames to estimate the prior for the current frame. Some systems assume no significant change in the camera pose between consecutive frames, and hence they assign the prior for the pose of the current frame to be the same as the previously tracked one.

The pose of the prior frame is used to guide the data association procedure in several ways. It helps determine a potentially visible set of features from the map in the current frame, thereby reducing the computational expense of blindly projecting the entire map. Furthermore, it helps establish an estimated feature location in the current frame, such that feature matching takes place in small search regions, instead of across the entire image. Finally, it serves as a starting point for the minimization procedure, which refines the camera pose.

2.4.2. Pose optimization
Direct and feature-based methods estimate the camera pose by minimizing a measure of error between frames; direct methods measure the photometric error, modeled as the intensity difference between pixels; in contrast, feature-based methods measure the re-projection error of landmarks from the map over the frame's prior pose. The re-projection error is formulated as the distance in pixels between a projected 3D landmark onto a frame, and its corresponding 2-D position in the image. A motion model is used to seed the new frame's pose at C_m (Fig. 3-H), and a list of potentially visible 3D landmarks from the map are projected onto the new frame. Data association takes place in a search window S_w surrounding the location of the projected landmarks. KSLAM then proceeds by minimizing an error vector (the geometric distance d in the case of feature-based methods or the intensity residuals in the case of direct methods) over the parameters of the rigid body transformation. To gain robustness against outliers, the minimization takes place over an objective function (Huber norm) that penalizes features with large errors. The camera pose optimization problem is then defined as:

$$T_i = \underset{T_i}{\arg\min} \sum_j Obj(e_j), \tag{3}$$

where T_i is a minimally represented Lie group of either $S\xi(3)$ or $sim(3)$ camera pose, $Obj(.)$ is an objective function and e_j is the error defined through data association for every matched feature j in the image. Finally, the system decides whether the new frame should be flagged as a keyframe or not. The decisive criteria can be categorized as either *significant pose change* or *significant scene appearance change*; a decision is usually made through a weighted combination of different criteria; examples of such criteria include: a significant change in the camera pose measurements (rotation and/or translation), the presence of a significant number of 2D features that are not observed in the map, a significant change in what the frame is observing (by monitoring the intensity histograms or optical flow), the elapsed time since the system flagged its latest keyframe, etc.

2.5. Topological/metric map generation

The map generation module is responsible for generating a representation of the previously unexplored, newly observed environment. Typically, the map generation module represents the world as a dense (for direct) or sparse (for feature-based) cloud of points. Fig. 6 presents the flowchart of a map generation module: different viewpoints of an unexplored scene are registered with their
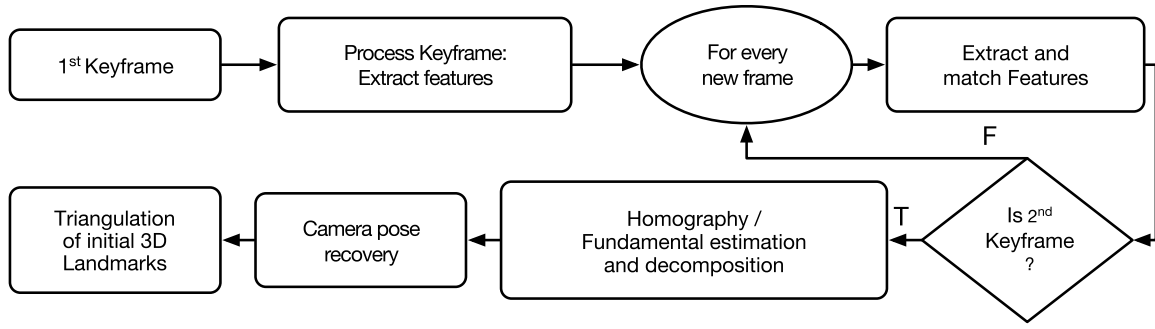
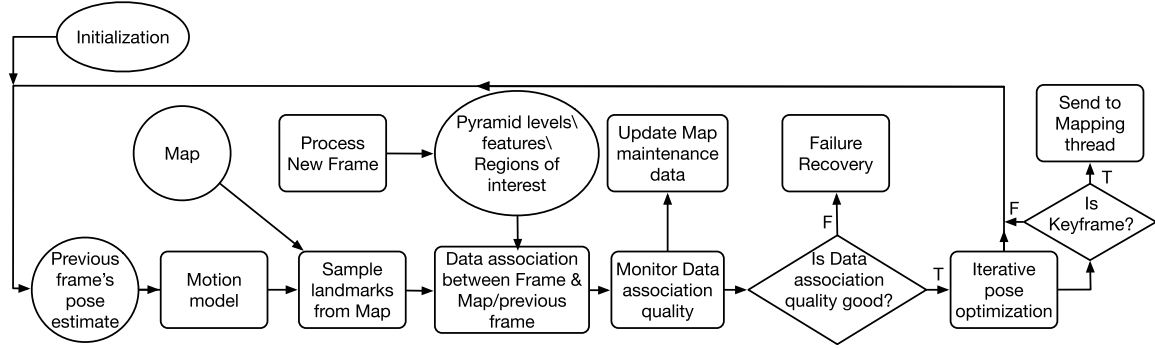**Fig. 4.** Generic model-based initialization flowchart.

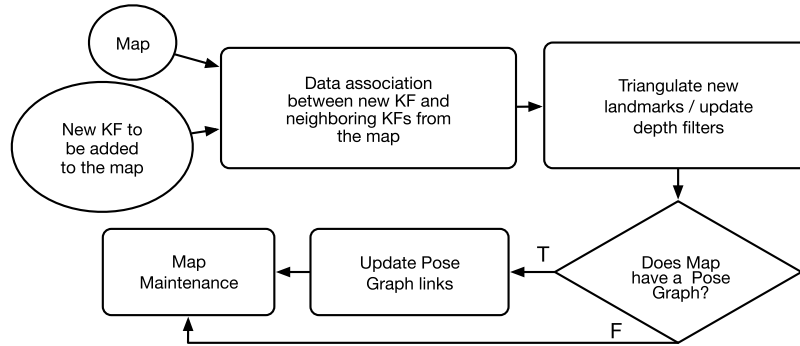**Fig. 5.** Generic pose estimation flowchart.

Fig. 6. Generic map generation flowchart.

corresponding camera poses through the pose tracking module. The map generation module then re-establishes data association between the new keyframe and a set of keyframes surrounding it, looking for matches. It then triangulates 2D points of interest into 3D landmarks as depicted in Fig. 3-I and J; it also keeps track of their 3D coordinates, and expands the map within what is referred to as a *metric* representation of the scene.

Topological maps were introduced to alleviate the computational cost associated with processing a global metric representation, by forfeiting geometric information in favor for connectivity information. In its most simplified form, a topological map consists of nodes corresponding to locations, and edges corresponding to connections between the locations. In the context of monocular SLAM, a topological map is an undirected graph of nodes that typically represents keyframes linked together by edges, when shared data associations between the keyframes exists, as depicted in Fig. 3-K. For a survey on topological maps, the reader is referred to [36]. In spite of the appeal of topological maps in scaling well with large scenes, metric information is still required in order to

maintain camera pose estimates. The conversion from a topological to a metric map is not always trivial, and for this reason, recent monocular SLAM systems [37–40] employ hybrid maps, which are locally metric and globally topological. The implementation of a hybrid map representation permits the system to first reason about the world at a high level, which allows for efficient solutions to loop closures and failure recovery using topological information; and second, to increase efficiency of the metric pose estimate, by limiting the scope of the map to a local region surrounding the camera [41]. A hybrid map allows for local optimization of the metric map, while maintaining scalability of the optimization over the global topological map [42].

2.5.1. Metric maps

In a metric map, the structure of new 3D landmarks is recovered, given the pose transformation relating two keyframes observing the landmark, from epipolar geometry using the corresponding data associations between the keyframes [43].

Due to noise in data association and pose estimates of the tracked images, projecting rays from two associated features will
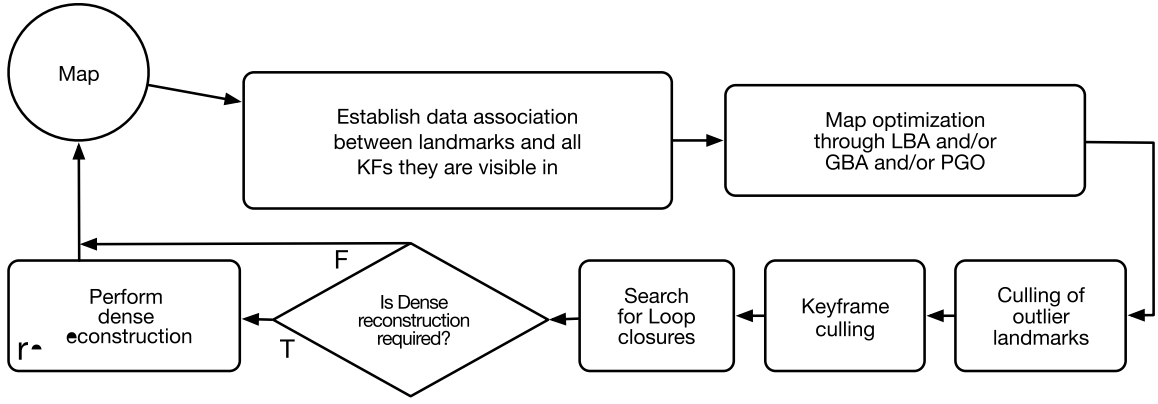
Fig. 7. Generic map maintenance flowchart.

most probably not intersect in 3D space. To gain resilience against outliers and to obtain better accuracy, the triangulation is typically performed over features associated across more than two views. Triangulation by optimization aims to estimate a landmark position $[x, y, z]$ from its associated 2D features across n views, by minimizing the sum of its re-projection errors in all keyframes observing it as described by:

$$X = \underset{[x,y,z]}{\operatorname{argmin}} \sum_n e_n,\qquad(4)$$

Filter based landmark triangulation recovers the 3D position of a landmark by first projecting into the 3D space a ray joining the camera center of the first keyframe observing the 2D feature and its associated 2D coordinates. The projected ray is then populated with a filter having a uniform distribution (D_1) of landmark position estimates, which are then updated as the landmark is observed across multiple views. The Bayesian inference framework continues until the filter converges from a uniform distribution to a Gaussian featuring a small variance (D_3) [44]. Filter-based triangulation results in a delay before an observed landmark's depth has fully converged, and can be used for pose tracking. To overcome this delay [45] suggested an inverse depth parametrization for newly observed features, with an associated variance that allows for 2D features to contribute to the camera pose estimate, as soon as they are observed.

2.5.2. Topological maps

When a new keyframe is added into systems that employ hybrid maps, their topological map is updated by incorporating the new keyframe as a node, and searching for data associations between the newly added node and surrounding ones; edges are then established to other nodes (keyframes) according to the number of found data associations: the thickness of the edges connecting the nodes is proportional to the number of common landmarks observed in those nodes.

2.6. BA/PGO/map maintenance

Map maintenance takes care of optimizing the map through either bundle adjustment or pose graph optimization [46]. Fig. 7 presents the steps required for map maintenance of a generic monocular SLAM: during map exploration, new 3D landmarks are triangulated based on the camera pose estimates; after some time, system drift manifests itself in wrong camera pose measurements, due to accumulated errors in previous camera poses that were used to expand the map. Map maintenance proceeds by establishing data association between the entire set of keyframes in the map or a subset of keyframes and performs a global bundle adjustment

(GBA) or a local bundle adjustment (LBA) respectively. Outlier landmarks flagged from the optimization are then culled (removed from the map). To reduce the complexity of the optimization, redundant keyframes are also culled. Map maintenance is also responsible for detecting and optimizing loop closures as well as performing a dense map reconstruction for systems that allow for it.

Bundle adjustment is the problem of refining a visual reconstruction to produce a jointly optimal 3D structure and viewing parameter estimates (camera pose and/or calibration). What we mean by this is that the parameter estimates are found by minimizing some cost function that quantifies the model fitting error, and that the solution is simultaneously optimal with respect to both structure and camera variations [47]. BA, depicted in Fig. 3-L. is an optimization that minimizes the cost function defined by:

$$\underset{T,X}{\operatorname{argmin}} \sum_{i=1}^{N} \sum_{j \in S_i} Obj(e(T_i, X_j)),\qquad(5)$$

where T_i is a keyframe pose estimate and N is the number of keyframes in the map or a subset of the map. X_j corresponds to the 3D pose of a landmark and S_i represents the set of 3D landmarks observed in Keyframe i. Finally, $e(T_i, X_j)$ is the re-projection error of a landmark X_j on a keyframe T_i, in which it is observed.

Bundle adjustment is computationally involved and intractable if performed on all frames and all poses. The breakthrough that enabled its application in PTAM is the notion of keyframes, where, only select frames labeled as keyframes, are used in the map creation process. Different algorithms apply different criteria for keyframe labeling, as well as different strategies for BA. Some perform a Global Bundle Adjustment (GBA), over the entire map. Others argue that a local BA only is sufficient to maintain a good quality map as such perform a Local Bundle Adjustment (LBA), over a local number of keyframes (also known as windowed optimization);

To reduce the computational expenses of bundle adjustment, [48] proposed to represent the monocular SLAM map by both a Euclidean map for LBA, and a topological map for pose graph optimization that explicitly distributes the accumulated drift along the entire map. PGO is best described by:

$$\underset{T}{\operatorname{argmin}} \sum_{i=1}^{N} \sum_{j \in S_i} Obj(e(T_i, X_j)).\qquad(6)$$

where the optimization process take place over the keyframe poses only (T_i).

Fig. 8 shows the map maintenance effect, where the scene's map is refined through outlier removal and error minimizations, in order to yield a more accurate scene representation.
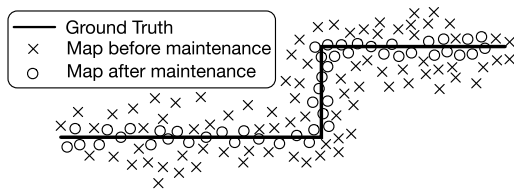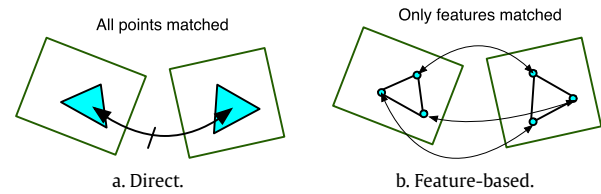
Fig. 8. Map maintenance effects on the map.

Fig. 9. Data association design choices: Direct vs. feature based methods.

2.7. Global localization

Global localization is required when the camera loses track of its position and is required to situate itself in a global map. Failure recovery and loop closure are both considered a form of global localization. It is noteworthy to mention that loop closure and failure recovery revolve around the same problem and solutions put forward to any of them could be used for the other. The interested reader is referred to [49] for a detailed survey on the topic. While there is no generic strategy to handle the global localization module, its various implementations are further discussed in Section 3. For now, we will limit the extent of the discussion to the problems global localization attempts to address.

2.7.1. Failure recovery

Whether due to wrong user movement, such as abrupt changes in the camera pose resulting in motion blur, or due to observing a featureless region, or for any other reason, monocular SLAM methods may eventually fail. An example of monocular SLAM failure is shown in Fig. 3-M, where due so sudden motion, the previously observed scene went out of view. Accordingly, a key module essential for the usability of any monocular SLAM system is its ability to correctly recover from such failures.

2.7.2. Loop closure

Since keyframe-based monocular SLAM is an optimization problem, it is prone to drifts in camera pose estimates. Returning to a certain pose after an exploration phase may not yield the same camera pose measurement, as it was at the start of the run. (See Fig. 3-N). Such camera pose drift can also manifest itself in a map scale drift, which will eventually lead the system to erroneous measurements, and fatal failure. To address this issue, some algorithms detect loop closures in an online monocular SLAM session, and optimize the loops track in an effort to correct the drift and the error in the camera pose and in all relevant map data that were created during the loop. The loop closure thread attempts to establish loops upon the insertion of a new keyframe, in order to correct and minimize any accumulated drift by the system over time using either PGO or BA; the implementations of such optimizations has been made easier using libraries such as G2o [46] and Ceres [50].

3. Design choices

Now that we have established a generic framework for KSLAM, this section details the design choices, made by different KSLAM systems in the literature. Table 1 lists all the KSLAM systems that, to our knowledge, exist to date; they are categorized as being either open-source or closed-source. Given the additional insight we gained by having access to their code, we will delve deeper into the open-source systems than we will in those that are closed-source. While this section will be devoted to the details of open-source systems, including PTAM, SVO, DT SLAM, LSD SLAM, ORB SLAM, DPPTAM and DSO; the closed source systems will be touched upon in Section 4.

3.1. Data association

3.1.1. Data association design choices

Table 2 summarizes the design choices for the data association used by open source KSLAM systems; **PTAM** and **DT SLAM** use FAST features [70] associated with a local patch of pixels as descriptors; **ORB SLAM** uses ORB features with associated ORB descriptors [23]. These are considered filter-based, as shown in Fig. 9b; in contrast, direct methods, that adopts the data association design of Fig. 9a, such as **LSD SLAM** and **DPPTAM**, extract and make use of *all* pixels that have a photometric gradient. **DSO** argues that using all the pixel information with a photometric gradient introduces redundancy in the system, and requires a regularization step; therefore, it suggests to subsample the pixels by dividing the image into blocks, keeping a fixed number of pixels with the highest gradient in each block. This ensures that first, the sampled pixels are well distributed across the image, and second, the sampled pixels have sufficiently high image gradients with respect to their immediate surroundings. The sampled pixels are referred to as candidate points. Different than other systems, **SVO** employs a hybrid approach in which it sequentially alternates between direct and feature-based methods.

3.1.2. Data association types:

Table 3 summarizes the feature extractors and their corresponding descriptors employed by various open-source KSLAM systems.

PTAM generates a 4 level pyramid representation of every incoming frame, as shown in Fig. 10, and uses it to enhance the features robustness to scale changes, and to increase the convergence radius of the pose estimation module. FAST features are extracted at each level with a Shi–Tomasi score [12] for each feature is estimated as a measure of the feature's saliency. Features with a relatively smaller score are removed before non-maximum suppression takes place. Once 2D features are extracted, 3D landmarks are projected onto the new frame, using a pose estimate prior (from motion model). 3D–2D data association is then employed. The descriptor used for data association is extracted from the 2D image from which the 3D landmark was first observed. To take into account viewpoint changes, the landmark's local patch of pixels descriptor is warped through an affine projection, which simulates how it would appear in the current frame. This however constitutes a limitation in PTAM, since, for large changes in camera viewpoints, the warping transform fails to accurately reflect the correct distortion, thereby causing data association failure.

When processing a new frame T_i, **DT SLAM** estimates a 2D similarity transform through image registration with the previous frame T_{i-1}, and transforms, using the estimated 2D similarity, features extracted from T_i into T_{i-1}. 3D landmarks are then projected onto T_{i-1} and data association takes place, similar to how it is done in PTAM. DT SLAM also keeps track of 2D landmarks, which are features that were previously observed but were not triangulated into 3D landmarks due to the lack of parallax between the different frames observing them (i.e., when the camera undergoes a pure

Table 1
Keyframe-based visual SLAM systems, with seven open-source, and sixteen closed-source systems.

| Year	Name	Closed/ open	Reference
2006	Real-time Localization and 3D Reconstruction	closed	[51]
2007	Parallel Tracking and Mapping (PTAM)	open	[52]
2008	An Efficient Direct Approach to Visual SLAM	closed	[53]
2010	Scale Drift-Aware Large Scale Monocular SLAM	closed	[54]
2010	Live dense reconstruction with a single moving camera	closed	[55]
2011	Dense Tracking and Mapping in Real-Time(DTAM)	closed	[56]
2011	Omnidirectional dense large-scale mapping and navigation based on meaningful triangulation	closed	[57]
2011	Continuous localization and mapping in a dynamic world (CD SLAM)	closed	[58]
2011	Online environment mapping	closed	[39]
2011	Homography-based planar mapping and tracking for mobile phones	closed	[59]
2013	Robust monocular SLAM in Dynamic environments (RD SLAM)	closed	[60]
2013	Handling pure camera rotation in keyframe-based SLAM (Hybrid SLAM)	closed	[61]
2014	Efficient keyframe-based real-time camera tracking	closed	[62]
2014	Semi-direct Visual Odometry (SVO)	open	[63]
2014	Large Scale Direct monocular SLAM (LSD SLAM)	open	[38]
2014	Deferred Triangulation SLAM (DT SLAM)	open	[64]
2014	Real-Time 6-DOF Monocular Visual SLAM in a Large Scale Environment	closed	[40]
2015	Robust large scale monocular Visual SLAM	closed	[65]
2015	ORB SLAM	open	[37]
2015	Dense Piecewise Parallel Tracking and Mapping (DPPTAM)	open	[66]
2016	Multi-level mapping: Real-time dense monocular SLAM	closed	[67]
2016	Robust Keyframe-based Monocular SLAM for Augmented Reality	closed	[68]
2016	Direct Sparse Odometry (DSO)	open	[69]

Table 2
Method used by different monocular SLAM systems. Abbreviations used: indirect (i), direct (d), and hybrid (h).

System	PTAM	SVO	DT SLAM	LSD SLAM	ORB SLAM	DPPTAM	DSO
Method	I	H	I	D	I	D	D

Table 3
Feature extractors and descriptors. Abbreviations:local patch of pixels (L.P.P.).

System	PTAM	SVO	DT SLAM	LSD SLAM	ORB SLAM	DPPTAM	DSO
Feature type	FAST	FAST	FAST	Intensity gradient	FAST	Intensity gradient	Intensity gradient
Feature descriptor	L.P.P.	L.P.P.	L.P.P.	L.P.P.	ORB	L.P.P.	L.P.P.
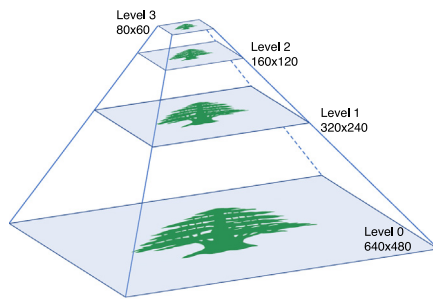
Fig. 10. A 4-level pyramid representation of an image. Each level is generated by blurring the level before it, and sub-sampling it by a factor of 2.

rotation motion). For every 2D landmark, the Euclidean distance between its epipolar line and the transformed feature is estimated; if it falls below a threshold, the feature is considered as a potential match to the 2D landmark. Data association through Zero Mean Sum of Squared Distance (ZMSSD) is then attempted to validate the matches.

SVO generates a five level pyramid representation of the incoming frame; data association is first established through iterative direct image alignment, starting from the highest pyramid level up until the third level. Preliminary data association from this step is used as a prior to a FAST feature matching procedure, similar to PTAM's warping technique, with a Zero-Mean SSD score.

ORB SLAM extracts FAST corners throughout eight pyramid levels. To ensure a homogeneous distribution along the entire image, each pyramid level is divided into cells and the parameters of the FAST detector are tuned online to ensure a minimum of five corners are extracted per cell. A 256-bit ORB descriptor is then computed for each extracted feature. ORB SLAM discretizes and stores the descriptors into bags of words, known as visual vocabulary [30], which are used to speed up image and feature matching by constraining those features that belong to the same node in the vocabulary tree. To deal with viewpoint changes, ORB SLAM proposes to keep track of all the keyframes in which a landmark is observed and choose the descriptor from the keyframe that has the smallest viewpoint difference with the current frame.

DSO Candidate points, sampled across the image, are represented by eight pixels spread around the target point. DSO claims that using this number of pixels in a specific pattern was empirically found to return a good trade-off between three objectives: computational time, sufficient information for tracking to take place, and resilience to motion blur. Each of the selected pixels around the candidate point contributes to the energy functional, which it seeks to minimize during tracking. Within this formulation, data association is still inherent from the direct image alignment scheme; however, using only the candidate points and their selected surrounding pixels, as opposed to using all pixels with gradients in an image. The added value this approach has over regular keypoint detectors is its adaptive ability to sample candidate points in low textured regions of the image.

3.2. Visual initialization: Choices made

The following section discusses the choices made for the initialization of KSLAM in light of Fig. 11, which graphically depicts the
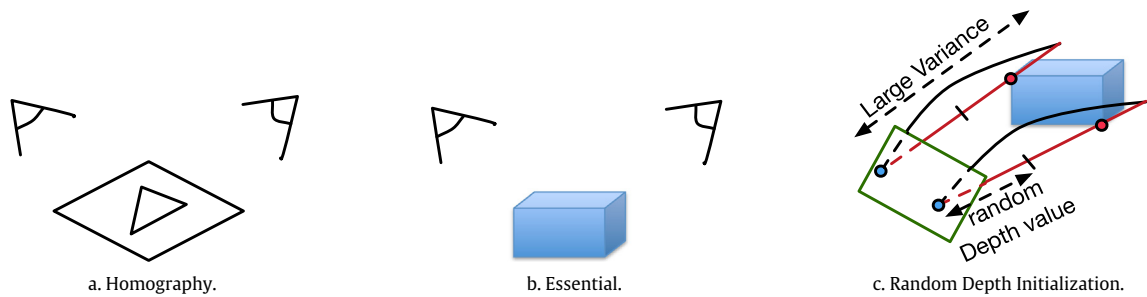
a. Homography. b. Essential. c. Random Depth Initialization.

Fig. 11. Different initialization methods used in KSLAM. (a) Homographies assume the scene to be planar; (b) The Essential matrix assumes the scene to be non-planar. (c) Random depth initialization, assign a random depth value to the first keyframe and *hope* they converge to a correct configuration in subsequent frames.

Table 4
Initialization. Abbreviations used: Homography decomposition (h.d.), Essential decomposition (e.d.), Random depth initialization (r.d.), Planar (p), Non-planar (n.p.), No assumption (n.a.).

System	PTAM	SVO	DT SLAM	LSD SLAM	ORB SLAM	DPPTAM	DSO
Initialization	h.d.	h.d.	e.d.	r.d.	h.d.+e.d.	r.d.	r.d.
Initial scene assumption	p	p	n.p.	n.a.	n.a.	n.a.	n.a.

different initialization options. Table 4 summarizes the initialization methods employed by different open source KSLAM systems, along with their association assumption of the observed scene at startup.

PTAM's [52] initial release suggested using the five-point algorithm [71] to estimate and decompose a Fundamental matrix into an $SE(3)$ transformation relating both initializing keyframes; the transformation is then used to triangulate an assumed non-planar initial scene. PTAM's initialization was later changed to the usage of a Homography [72], where the scene is assumed to be composed of 2D planes. PTAM's initialization requires the user's input to capture the first two keyframes in the map; furthermore, it requires the user to perform, in between the first and the second keyframe, a slow, smooth and relatively significant translational motion parallel to the observed scene. As the 2D–2D matching procedure takes place via ZMSSD without warping the features, establishing correct matches is susceptible to both motion blur, and significant appearance changes of features as a result of camera rotations; hence, the strict requirements on the user's motion during the initialization. The generated initial map is scaled such as the estimated translation between the first two keyframes corresponds to 0.1 units, before structure-only BA takes place.

SVO, [63] adopted a Homography for initialization with the same procedure as PTAM; SVO extracts FAST features and tracks them using KLT (Kanade–Lucas–Tomasi feature tracker) [73] across incoming frames. To avoid the need for a second input by the user, SVO monitors the median of the baseline distance of the features, tracked between the first keyframe and the current frame; and whenever this value reaches a certain threshold, sufficient parallax is assumed, and the Homography can be estimated.

DT SLAM does not have an explicit initialization phase; rather, it is integrated within its tracking module as an Essential matrix estimation method.

[38] suggested in **LSD SLAM**, and later in **DSO**, a randomly initialized scene's depth from the first viewpoint, Both systems use an initialization method that does not require two view geometry; it takes place on a single frame: pixels of interest (*i.e.*, image locations that have high intensity gradients) in the first keyframe are given a random depth value with an associated large variance. This results in an initially erroneous 3D map. The pose estimation methods are then invoked to estimate the pose of newly incoming frames using the erroneous map, which in return results in erroneous pose estimates. However, as the system process more frames of the same scene, the originally erroneous depth map converges to a
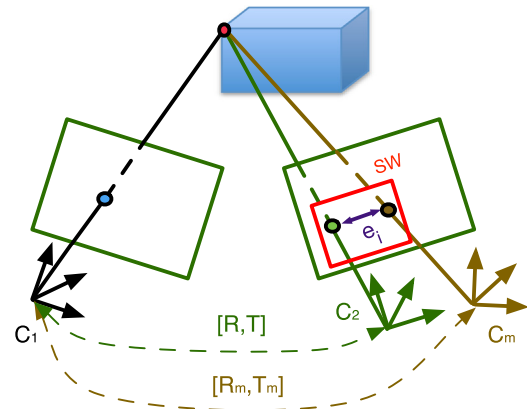

Fig. 12. A generic depiction of the pose estimation module: given a previous pose and a map, the frame of the new pose is first *guessed* using a motion model C_m. The motion model bounds the data association into a search window SW. An error vector e_i (either geometric or photometric) is then computed and minimized over the initial guess C_m until it converges to the actual pose C_2.

stable solution. The initialization is considered complete when the depth variance of the initial scene converges to a minimum.

DPPTAM, [66] borrows from LSD SLAM's initialization procedure, and therefore also suffers from the problem of random depth initialization, where several keyframes must be added to the system before a stable configuration is reached.

ORB SLAM deals with the limitations arising from all the above methods by computing, in parallel, both a Fundamental matrix and a Homography [37]; in order to select the appropriate model, each model is penalized according to its symmetric transfer error [35]. If the chosen model yields poor tracking quality, and too few feature correspondences in the upcoming frame, the initialization is discarded, and the system restarts with a different pair of frames.

3.3. Pose estimation: Choices made

This section discusses the details of the pose estimation module, as proposed by all open source KSLAM systems, in light of the generic pose estimation process depicted in Fig. 12. Table 5 summarizes the pose estimation methods used by different monocular SLAM systems.

Table 5
Pose estimation. Abbreviations are as follows: constant velocity motion model (c.v.m.m), same as previous pose (s.a.p.p.), similarity transform with previous frame (s.t.p.f.), optimization through minimization of geometric error(o.m.g.e.), optimization through minimization of photometric error (o.m.p.e.), Essential matrix decomposition (E.m.d.), pure rotation estimation from 2 points (p.r.e.), significant pose change (s.p.c.), significant scene appearance change (s.s.a.c.).

System	PTAM	SVO	DT SLAM	LSD SLAM	ORB SLAM	DPPTAM	DSO
Motion prior	c.v.m.m. +ESM	s.a.p.p.	s.t.p.f.	s.a.p.p.	c.v.m.m. or place recogn.	c.v.m.m. or s.a.p.p.	multiple c.v.m.m.
Tracking	o.m.g.e.	o.m.p.e.	e.m.d. or o.m.g.e. or p.r.e.	o.m.p.e.	o.m.g.e.	o.m.p.e.	o.m.p.e. and o.m.g.e.
keyframe add criterion	s.p.c.	s.p.c.	s.s.a.c.	s.p.c.	s.s.a.c.	s.p.c.	s.s.a.c. or s.p.c.

PTAM defines the camera pose C as an $SE(3)$ transformation that can be minimally represented by six parameters. The mapping from the full $SE(3)$ transform to its minimal representation $S\xi(3)$ and vice versa can be done through logarithmic and exponential mapping in Lie algebra [74]. The minimally represented $S\xi(3)$ transform is of great importance as it reduces the number of parameters to optimize from twelve to six, leading to significant speedups in the optimization process.

In PTAM, pose estimation starts by estimating a prior to the frame's pose using a decaying constant velocity motion model. The prior is then refined using a Small Blurry Image (SBI) – the smallest image resolution in the pyramid representation of the frame – by applying an *Efficient Second Order minimization* [75]. If the velocity is high, PTAM anticipates a fast motion is taking place, and hence, the presence of motion blur and thus restricts tracking to take place only at the highest pyramid levels (most resilient to motion blur) in what is referred to as a coarse tracking stage. Otherwise, the coarse tracking stage is followed by a fine tracking stage. However, when the camera is stationary, the coarse stage may lead to jittering of the camera's pose, and is therefore turned off.

The initial camera pose prior is then refined by minimizing a Tukey-biweight [76] objective function of the re-projection error that down-weights observations with large residuals. To determine the tracking quality, PTAM monitors the ratio of successfully matched features in the frame, against the total number of attempted feature matches. **SVO** assumes the pose of the new frame to be the same as the previous one; it then searches for the transformation that minimizes the photometric error of the image pixels with associated depth measurements in the current frame, with respect to their location in the previous one. The minimization takes places through thirty Gauss–Newton iterations of the inverse compositional image alignment method.

SVO does not employ explicit feature matching for every incoming frame; rather, it is performed implicitly as a byproduct of the image alignment step. Once image alignment takes place, landmarks that are expected to be visible in the current frame, are projected onto the image. To decrease the computational complexity and to maintain only the strongest features, the frame is divided into a grid, and only the strongest feature per grid cell is used. The 2D location of the projected landmark is fine-tuned by minimizing the photometric error between its associated patch from its location in the current frame, and a warp of the landmark generated from the nearest keyframe observing it. This minimization violates the epipolar constraint for the entire frame, and further processing in the tracking module is required: motion-only bundle adjustment takes place, followed by a structure only bundle adjustment that refines the 3D location of the landmarks, based on the refined camera pose. Finally, a joint (pose and structure) local bundle adjustment fine-tunes the reported camera pose estimate. During this last stage, the tracking quality is continuously monitored and, if the number of observations in a frame, or the number of features between consecutive frames drop, tracking quality is deemed insufficient, and failure recovery methods are initiated.

DT SLAM maintains a camera pose based on three tracking modes: full pose estimation, Essential matrix estimation, and pure rotation estimation. When a sufficient number of 3D matches exists, a full pose can be estimated; otherwise, if a sufficient number of 2D matches that exhibit small translations is established, an Essential matrix is estimated; and finally, if a pure rotation is exhibited, two points are used to estimate the absolute orientation of the matches [77]. Pose estimation aims, in an iterative manner, to minimize the error vector of both 3D–2D re-projections, and 2D–2D matches. When tracking failure occurs, the system initializes a new map and continues to collect data for tracking in a different map; however, the map making thread continues to look for possible matches between the keyframes of the new map and the old one, and once a match is established, both maps are fused together, thereby allowing the system to handle multiple sub-maps, each at a different scale.

The tracking thread in **LSD SLAM** is responsible for estimating the pose of the current frame with respect to the currently active keyframe in the map, using the previous frame pose as a prior. The required pose is represented by an $SE(3)$ transformation, and is found by an iteratively re-weighted Gauss–Newton optimization that minimizes the variance normalized photometric residual error, as described in [78]. A keyframe is considered active if it is the most recent keyframe accommodated in the map. To minimize outlier effects, measurements with large residuals are down-weighted from one iteration to the next.

Pose estimation in **ORB SLAM** is established through a constant velocity motion model prior, followed by a pose refinement using optimization. As the motion model is expected to be easily violated through abrupt motions, ORB SLAM detects such failures by tracking the number of matched features; if it falls below a certain threshold, map points are projected onto the current frame, and a wide-range feature search takes place around the projected locations.

In an effort to make ORB SLAM operate in large environments, a subset of the global map, known as the local map, is defined by all landmarks corresponding to the set of all keyframes that share edges with the current frame, as well as all neighbors of this set of keyframes from the pose graph. The selected landmarks are filtered out to keep only the features that are most likely to be matched in the current frame. Furthermore, if the distance from the camera's center to the landmark is beyond the range of the valid features, the landmark is also discarded. The remaining set of landmarks is then searched for and matched in the current frame, before a final camera pose refinement step .

Similar to LSD SLAM, **DPPTAM** optimizes the photometric error of high gradient pixel locations between two images, using the ICIA formulation over the $SE(3)$ transform relating them. The minimization is started using a constant velocity motion model, unless the photometric error increases after applying it. If the latter is true, the motion model is disregarded, and the pose of the last tracked frame is used. Similar to PTAM, the optimization takes place in the tangent space $S\xi(3)$ that minimally parameterizes the rigid body transform by six parameters.

DSO tracking and mapping threads are intertwined; in DSO, all frames are simultaneously tracked, and used in the map update process; however, each frame contributes differently, and is treated according to whether it is considered a keyframe or not.

DSO uses two parallel threads: a front-end thread, and a mapping thread. This section will further elaborate on the front-end thread, whereas the mapping thread will be detailed in Section 3.4.

DSO front-end initializes the system at startup using random-depth initialization; it computes the intensity gradients, and tracks the current frame with respect to the currently active keyframe. Different than other systems, DSO does not use a single frame pose prior; rather, it attempts a direct image alignment by looping over multiple pose guesses, in a pyramidal implementation, and removes guesses that yield higher residuals between iterations. The final pose estimate that yields the smallest residual error is then assigned to the current frame. The list of initial pose guesses includes:

- A constant velocity motion model (CVMM),
- A motion model that assumes twice the motion of the CVMM guess,
- Half of the CVMM guess,
- No motion at all (use the pose of the active keyframe) and
- Randomly selected small rotations surrounding the CVMM guess.

Finally, the front-end checks if the frame should be a keyframe based on one of the following three conditions: 1st when the field of view between the current frame and the last observed keyframe has changed significantly; 2nd when the camera undergoes a significant amount of translation; and 3rd if the relative brightness factor between the two frames changes significantly. A weighted mixture between these conditions is used, and compared to a threshold to decide whether a frame becomes a keyframe or not. On average, around five to ten keyframes are added per second. If desired, one can manually select the rate of acceptance of keyframe, and disable any of the above selection criteria.

3.4. Topological/metric map generation: Choices made

This section discusses the details of the map generation module, as proposed by all open source KSLAM systems, in light of the generic map expansion process depicted in Fig. 13. Table 6 summarizes map generation methods employed by different monocular SLAM systems.

When a new keyframe is added in **PTAM**, all bundle adjustment operations are halted, and the new keyframe inherits the pose from the coarse tracking stage. The potentially visible set of landmarks estimated by the tracker are then re-projected onto the new keyframe, and feature matches are established. Correctly matched landmarks are marked as seen again; this is done to keep track of the quality of the landmarks and to allow for the map refinement step to remove corrupt data.

New landmarks are generated by establishing and triangulating feature matches between the newly added keyframe and its nearest keyframe (in terms of position) from the map. Landmarks that are already existent in the map are projected onto both keyframes, and feature matches from the current keyframe are searched for along their corresponding epipolar lines in the second keyframe, at regions that do not contain projected landmarks. The average depth of the projected landmarks is used to constrain the epipolar search, from a line to a segment.

SVO parametrizes 3D landmarks using an inverse depth parametrization model [45]. Upon insertion of a new keyframe, features possessing the highest Shi–Tomasi scores are chosen to initialize a number of depth filters. These features are referred to as seeds, and are initialized along a line propagating from the camera center to the 2D location of the seed in the originating keyframe. The only parameter that remains to be solved for is then the depth of the landmark, which is initialized to the mean of the scene's depth, as observed from the keyframe of origin.

During the times when no new keyframe is being processed, the map management thread monitors and updates map seeds by subsequent observations, in a fashion similar to [79]. The seed is searched for in new frames along an epipolar search line, which is limited by the uncertainty of the seed, and the mean depth distribution observed in the current frame. As the filter converges, its uncertainty decreases, and the epipolar search range decreases. If seeds fail to match frequently, if they diverge to infinity, or if a long time has passed since their initialization, they are removed from the map. This process however limits SVO to operate in environments of relatively uniform depth distributions. Since the initialization of landmarks in SVO relies on many observations in order for the features to be triangulated, the map contains few, if any, outliers, and hence no outlier deletion method is required. However, this comes at the expense of a delayed time before the features are initialized as landmarks and added to the map.

Inspired by the map generation module of SVO, **REMODE** was released by [80] as a standalone map generation module that takes input measurements from SVO (Sparse map and pose estimates) and generates a per-pixel dense map of the observed scene, based on the probabilistic Bayesian scheme suggested in [79].

DT SLAM aims to add keyframes when enough visual change has occurred; the three criteria for keyframe addition are (1) for the frame to contain a sufficient number of new 2D features that can be created from areas not covered by the map, or (2) a minimum number of 2D features can be triangulated into 3D landmarks, or (3) a given number of already existing 3D landmarks have been observed from a significantly different angle. The map in DT SLAM contains both 2D and 3D landmarks, where the triangulation of 2D features into 3D landmarks is done through two view triangulation by optimization, and is deferred until enough parallax between the keyframes is observed—hence the name of the algorithm.

LSD SLAM's map generation module is mainly responsible for the selection and accommodation of new keyframes into the map. Its functions can be divided into two main categories, depending on whether the current frame is a keyframe or not; if it is, depth map creation takes place by keyframe accommodation as described below; if not, depth map refinement is done on regular frames.

When the system is accommodating a new keyframe, the estimated depth map from the previous keyframe is projected onto it, and serves as its initial guess. Spatial regularization then takes place, by replacing each projected depth value with the average of its surrounding values, and the variance is chosen as the minimal variance value of the neighboring measurements. The $Sim(3)$ of a newly added keyframe is then estimated and refined in a direct, scale-drift aware image alignment scheme with respect to other keyframes in the map, over the seven degree of freedom $Sim(3)$ transform. Due to the non-convexity of the direct image alignment method on $Sim(3)$, an accurate initialization to the minimization procedure is required; for such a purpose, ESM (Efficient Second Order minimization) [75] and a coarse to fine pyramidal scheme with very low resolutions proved to increase the convergence radius of the task.

ORB SLAM's local mapping thread is responsible for keyframe insertion, map point triangulation, map point culling, keyframe culling, and local bundle adjustment. ORB SLAM incorporates a hybrid, one metric and two topological maps. The two topological maps, referred to as co-visibility and essential graphs, are built using the same nodes (keyframes) however, with different edges (connections) between them. The co-visibility graph allows for as many connections as available between nodes; in contrast to the essential graph that allows every node to have at most two edges, by only keeping the strongest two edges. The difference between them is contrasted in Fig. 14.

The mapping thread is responsible for updating the co-visibility and essential graphs with the appropriate edges, as well as computing the bag of words representing the newly added keyframes
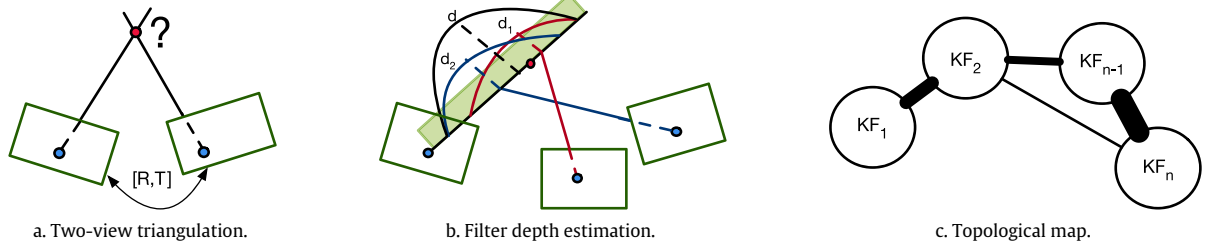
Fig. 13. A generic depiction of the different solutions employed to generate/expand a representation of the observed scene: (a) two-view triangulation represents a world surface point by a 3D vector $(x, y, z)^t$. (b) Filter depth estimation assumes a world surface point to exist along a semi-infinite ray going from the center of the first camera in which it was observed to infinity, passing by its corresponding 2 dimensional pixel projection. The depth is then found through subsequent filter updates from small baseline observations. (c) A Topological map is an undirected graph, that represent keyframes as nodes connected with an edge to other keyframes, with which they share a significant amount of data association.

Table 6
Map generation. Abbreviations: 2 view triangulation (2.v.t.), particle filter with inverse depth parametrization (p.f.), 2D landmarks triangulated to 3D landmarks (2D.l.t.), depth map propagation from previous frame (p.f.p.f.), depth map refined through small baseline observations (s.b.o.), multiple hypotheses photometric error minimization (m.h.p.m.).

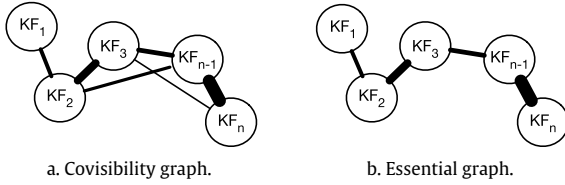| System	PTAM	SVO	DT SLAM	LSD SLAM	ORB SLAM	DPPTAM	DSO
Map generation	2.v.t.	p.f.	2D.l.t.	p.f.p.f and s.b.o.	2.v.t.	m.h.p.m	s.b.o.
Map type	metric	metric	metric	hybrid	hybrid	metric	metric

Fig. 14. The difference between a. co-visibility graph and b. essential graph. An essential graph is obtained by thresholding the co-visibility graph to keep the strongest two edges per keyframe.

in the map. The metric map is propagated by triangulating new landmarks from ORB features, which appear in at least two nodes connected to the new keyframe in the co-visibility graph. To prevent outliers, triangulated landmarks are tested for positive depth, re-projection error, and scale consistency in all keyframes they are observed in, before finally incorporating them into the map.

Landmark triangulation in **DPPTAM** takes place over several overlapping observations of the scene using inverse depth parametrization; the map maker aims to minimize the photometric error between a high gradient pixel patch in the last added keyframe, and the corresponding patch of pixels, found by projecting the feature from the keyframe onto the current frame. The minimization is repeated ten times for all high-gradient pixels, when the frame exhibits enough translation; the threshold for translation is increased from one iteration to the next, to ensure sufficient baseline distance between the frames. The end result is ten hypotheses for the depth of each high-gradient pixel. To deduce the final depth estimate from the hypotheses, three consecutive tests are performed, including gradient direction test, temporal consistency, and spatial consistency.

All frames in **DSO** are used in the map building process; while keyframes are used to expand the map and perform windowed optimization, regular (non-keyframe) frames are used to update the depth of the already existing candidate points. DSO maintains two thousand candidate points per keyframe. The estimated pose of subsequent regular frames, the location of the candidate points in the active keyframe and their variance, are all used to establish an epipolar search segment in the regular frame. The image location along the epipolar segment, which minimizes the photometric error, is used to update the depth and the variance of the candidate point, using a filter-based triangulation, similar to LSD SLAM. DSO adopts the inverse depth paradigm as a parametrization for the 3D world which reduces the parameters to optimize to one variable; thereby reducing computational cost. This estimated depth is used as a prior for a subsequently activated candidate point in a windowed optimization.

In its active window of optimization, DSO maintain seven *active keyframes*, along with two thousand *active points*, equally distributed across the active keyframes. As new keyframes and candidate points are accommodated by the system, older ones are marginalized: where the number of active keyframes exceeds 7, the system chooses a keyframe from the active window and marginalizes it. The choice of the keyframe is done by maximizing a heuristically designed distance score, which ensures the remaining active keyframes to be well distributed across the space between the first and last keyframes in the active window, and closer to the most recently added keyframe. Also if ninety five percent of a frame's points are marginalized, the frame is dropped out of the system.

3.5. BA/PGO/map maintenance: Choices made

As detailed in Section 2.6, there are many variations of the optimization process that can be applied to KSLAM, namely LBA (local bundle adjustment), GBA(global bundle adjustment), PGO (pose graph optimization), and structure only bundle adjustment. These different variations are shown in Fig. 15.

Table 7 summarizes the map maintenance procedure adopted by different KSLAM systems.

When the map making thread is not processing new keyframes, **PTAM** performs various optimizations and maintenance to the map, such as an LBA for local map convergence and a GBA for the map's global convergence. The computational cost in PTAM scales with the map and becomes intractable as the number of keyframes gets large; for this reason, PTAM is designed to operate in small workspaces. Finally, the optimization thread applies data refinement by first searching and updating landmark observations in all keyframes, and then by removing all landmarks that failed, many times, to successfully match features.

For runtime efficiency reasons, **SVO**'s map management maintains only a fixed number of keyframes in the map and removes
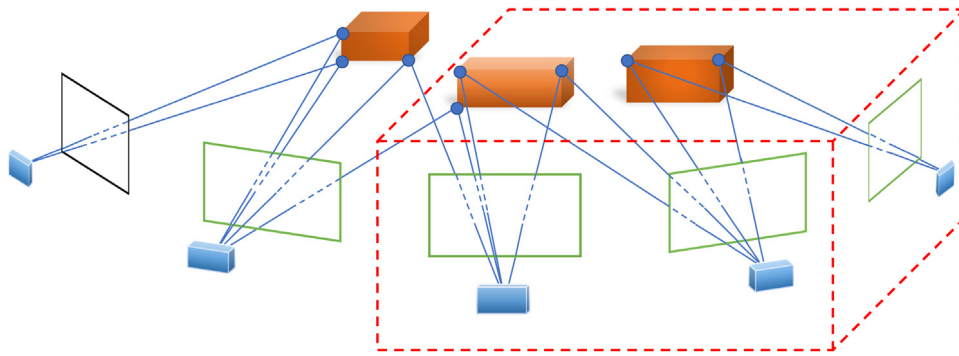
Fig. 15. Different optimization strategies: GBA jointly optimizes the entire map whereas LBA jointly optimizes a subset of the map (encapsulated in the red cube). PGO optimizes the poses of a subset of frames from the map (green frames). Structure only bundle adjustment optimizes the 3D position of the landmarks (blue circles) while holding their associated frames poses fixed. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 7
Map maintenance. Abbreviations used: Local Bundle Adjustment (LBA), Global Bundle Adjustment (GBA), Pose Graph Optimization (PGO), Structure Only Bundle Adjustment (SOBA).

System	PTAM	SVO	DT SLAM	LSD SLAM	ORB SLAM	DPPTAM	DSO
Optimization	LBA & GBA	LBA & SOBA	LBA &GBA	PGO	PGO & LBA	Dense mapping	LBA
Scene type	static & small	static & uniform depth	static & small	static & small or large	static & small or large	static & indoor planar	static & small or large

distant ones when new keyframes are added. This is performed so that the algorithm maintains real-time performance after prolonged periods of operation over large distances.

Aside from the map generation module, **DT SLAM** employs a third thread that continuously optimizes the entire map in the background through a sparse GBA.

LSD SLAM runs a third parallel thread that continuously optimizes the map in the background by a generic implementation of a pose graph optimization using the g2o-framework [46]. This however leads to an inferior accuracy when compared to other methods. Outliers are detected by monitoring the probability of the projected depth hypothesis at each pixel of being an outlier or not. To make the outliers detection step possible, LSD SLAM keeps records of each successfully matched pixel during the tracking thread, and accordingly increases or decreases the probability of it being an outlier. **ORB SLAM** employs rigorous landmark culling to ensure few outliers in the map. A landmark must be correctly matched to twenty five percent of the frames in which it is predicted to be visible. It also must be visible from at least three keyframes after more than one keyframe has been accommodated into the map, since it was spawned. Otherwise, the landmark is removed. To maintain lifelong operation and to counter the side effects of the presence of a high number of keyframes in the map, a rigorous keyframe culling procedure takes place as well. Keyframes that have ninety percent of their associated landmarks observed in three other keyframes are deemed redundant, and removed. The local mapping thread also performs a local bundle adjustment over all keyframes connected to the latest accommodated keyframe in the co-visibility graph, and all other keyframes that observe any landmark present in the current keyframe.

DPPTAM produces real-time dense maps by employing a dense mapping thread that exploits planar properties of man-made indoor environments. Keyframes are first segmented into a set of 2D superpixels, and all 3D landmarks from the map are projected onto the keyframe, and assigned to different superpixels according to the distance of their projections to the appropriate superpixel in the keyframe. 3D points belonging to contours of the superpixels are used to fit 3D planes to each superpixel. To determine if the superpixel's plane is to be added into the map, three test are performed, including the normalized residual test, the degenerate

case detection, and the temporal consistency test. Finally, a full dense map is reconstructed, by estimating depth at every pixel, using depth priors of the 3D planes associated with the superpixels.

Analogous to LBA, **DSO** performs a windowed optimization over the combined photometric (intensity) and geometric residual of all active points between the set of active keyframes, using six Gauss–Newton iterations. If the resulting residual of the most recently added keyframe after the optimization is large, the newly added keyframe is dropped. Map maintenance in DSO is also responsible for outlier detection and handling in an early stage of the system. The point is disregarded and dropped out if one of the four following conditions is satisfied:

1. A point cannot be distinctively identified from its immediate surroundings,
2. A point falls out of bound of the search region,
3. A point match is found farther than two pixels from its original location or
4. The point has a relatively high photometric residual with respect to the median residual of the entire frame.

3.6. Failure recovery: Choices made

Table 8 summarizes the failure recovery mechanisms used by different monocular SLAM system.

Upon detecting failure, **PTAM**'s tracker initiates a recovery procedure, where the SBI of each incoming frame is compared to the database of SBIs for all keyframes. If the intensity difference between the incoming frame and its nearest looking keyframe is below a certain threshold, the current frame's pose is assumed to be equivalent to that of the corresponding keyframe. ESM tracking takes place to estimate the rotational change between the keyframe and the current frame. If converged, the tracker attempts to match the landmarks to the features in the frame. If a sufficient number of features are correctly matched, the tracker resumes normally; otherwise, a new frame is acquired and the tracker remains lost.

SVO's first procedure in the recovery process is to apply image alignment between the incoming frame and the closest keyframe

Table 8
Failure recovery. Abbreviations used: photometric error minimization of SBIs (p.e.m.), image alignment with last correctly tracked keyframe (i.a.l.), image alignment with random keyframe (i.a.r.), bag of words place recognition (b.w.), image alignment with arbitrary small rotations around the last tracked pose(i.a.a.r.).

System	PTAM	SVO	DT SLAM	LSD SLAM	ORB SLAM	DPPTAM	DSO
Failure recovery	p.e.m.	i.a.l.	none	i.a.r.	b.w.	i.a.l	i.a.a.r.

to the last known correctly tracked frame. If more than thirty features are correctly matched during this image alignment step, then the re-localizer considers itself converged and continues tracking regularly; otherwise, it attempts to re-localize using new incoming frames.

LSD SLAM's recovery procedure first chooses, at random, from the pose graph, a keyframe that has more than two neighboring keyframes connected to it. It then attempts to align the currently lost frame to it. If the outlier-to-inlier ratio is large, the keyframe is discarded, and replaced by another keyframe at random; otherwise, all neighboring keyframes connected to it in the pose graph are then tested. If the number of neighbors with a large inlier-to-outlier ratio is larger than the number of neighbors with a large outlier-to-inlier ratio, or if there are more than five neighbors with a large inlier-to-outlier ratio, the neighboring keyframe with the largest ratio is set as the active keyframe, and regular tracking resumes.

Upon running, **ORB SLAM**'s re-localizer transforms the current frame into a bag of words and queries the database of keyframes for all possible keyframes that might be used to re-localize from. The place recognition module implemented in ORB SLAM, used for both loop detection and failure recovery, relies on bags of words, as frames observing the same scene share a big number of common visual vocabulary. In contrast to other bag of words methods that return the best queried hypothesis from the database of keyframes, the place recognition module of ORB SLAM returns all possible hypotheses that have a probability of being a match larger than seventy five percent of the best match. The combined added value of the ORB features, along with the bag of words implementation of the place recognition module, manifest themselves in a real-time, high recall, and relatively high tolerance to viewpoint changes during re-localization and loop detection. All hypotheses are then tested through a RANSAC implementation of the PnP algorithm [81], which determines the camera pose from a set of 3D to 2D correspondences. The camera pose with the most inliers is then used to establish more matches to features associated with the candidate keyframe, before an optimization over the camera's pose takes place.

DSO does not have a global based failure recovery method. When the minimization of DSO's pose tracking diverges, the last successfully tracked camera pose is used to generate multiple arbitrary random rotations around it. The generated poses are used in an attempt to localize at the coarsest pyramid level with the most recent active keyframe; if the photometric minimization succeeds, regular tracking resumes, otherwise, tracking fails.

3.7. Loop closure: Choices made

Table 9 summarizes how loop closure is addressed by each KSLAM system.

When tracking failure occurs in **DT SLAM**, it starts a new sub map and instantly start tracking it, while it invokes, in the background, a loop closure thread that attempts to establish data associations across different sub-maps. DT SLAM's loop closure module in this context is a modified version of PTAM's failure recovery module, but employed across sub maps. When a sufficient number of data associations are successfully established between two keyframes, their corresponding sub-maps are merged together through a similarity transform optimization.

Whenever a keyframe is processed by **LSD SLAM**, loop closures are searched for within its ten nearest keyframes as well as through the appearance based model of FABMAP [82] to establish both ends of a loop. Once a loop edge is detected, a pose graph optimization minimizes the similarity error established at the loop's edge, by distributing the error over the poses of the loop's keyframes.

Loop detection in **ORB SLAM** takes place via its global place recognition module, that returns all hypotheses of keyframes, from the database that might correspond to the opposing loop end. All landmarks associated with the queried keyframe and its neighbors are projected to, and searched for, in all keyframes associated with the current keyframe in the co-visibility graph. The initial set of inliers, as well as the found matches, are used to update the co-visibility and Essential graphs, thereby establishing many edges between the two ends of the loop. Finally, a pose graph optimization over the essential graph takes place, similar to that of LSD SLAM, which minimizes and distributes the loop closing error along the loop nodes.

Marginalized keyframes and points in **DSO** are permanently dropped out of the system and never used again; therefore, DSO does not employ any global based localization methods to recover from failure, or detect loop closures.

4. Closed source systems

We have discussed so far methods presented in open-source monocular SLAM systems; however, many closed source methods also exist in the literature that we could not fully dissect due to the limited details presented in their papers. This section aims to provide a quick chronological survey of these systems, which put forward many interesting ideas for the reader. To avoid repetition, we will not outline the complete details of each system; rather, we will focus on what we feel has additive value to the reader beyond the information provided in Section 3. For the remainder of this section, each paragraph is a concise summary of the main contributions in the closed-source systems.

In 2006, [51] were the first to introduce the concept of keyframes in monocular SLAM and employed a local Bundle Adjustment in real-time over a subset of keyframes in the map. To ensure sufficient baseline, the system is initialized by automatic insertion of three keyframes. The second keyframe is only used to ensure that there has been sufficient baseline between the first and the third.

[53] proposed a real-time direct solution by assuming large patches surrounding regions of high intensity gradients as planar; and then performing image alignment by minimizing the photometric error of these patches across incoming frames, in a single optimization step that incorporates Cheirality, geometric, and photometric constraints. The system employs a photogeometric generative model to gain resilience against lighting changes, and monitors the errors in the minimization process to flag outliers.

In 2009, [83] suggested, in Keyframe-Based Real-Time Camera Tracking, to perform monocular SLAM in two steps. The first step is an offline procedure during which, SIFT features are extracted and triangulated using SfM methods into 3D landmarks. The generated scene is then sub-sampled into keyframes according to a selection criterion that minimizes a redundancy term and a completeness term. To enhance the performance of subsequent steps, the keyframes descriptors are stored in a vocabulary tree.

Table 9
Loop closure. Abbreviations used: Bag of Words place recognition (B.W.p.r), sim(3) optimization (s.o.).

System	PTAM	SVO	DT SLAM	LSD SLAM	ORB SLAM	DPPTAM	DSO
Loop closure	none	none	none	FabMap +s.o.	B.W.p.r. +s.o.	none	none

During the online session, the system estimates a subset of keyframes from the map, using the vocabulary tree, with keyframes that share the most features with the current frame. It then employs a parallelized 2D–3D data association to establish an error vector between the landmarks observed in the keyframes and the current frame, which is used to estimate the current camera pose.

The system was then updated in 2014 in [62] to include a GPU accelerated SIFT feature extraction and matching, with a proposed two-pass feature matching procedure. In the first pass, SIFT features from the current frame are matched to the selected keyframes, using the offline-generated vocabulary tree. This is followed by an outlier rejection implementation through RANSAC; if the remaining number of inliers is below a threshold, the second pass is performed. During the second pass, the epipolar geometry, found from the first pass, is used to constrain local SIFT feature matching between the keyframes and the current frame. The process repeats until a sufficient number of inliers is found.

[54] introduced similarity transforms into monocular SLAM, allowing for scale drift estimation and correction, once the system detects loop closure. Feature tracking is performed by a mixture of top-bottom and bottom-up approaches, using a dense variational optical flow, and a search over a window surrounding the projected landmarks. Landmarks are triangulated by updating information filters, and loop detection is performed using a bag of words discretization of SURF features [19]. The loop is finally closed by applying a pose graph optimization over the similarity transforms relating the keyframes.

[55] suggested a hybrid monocular SLAM system that relied on feature-based SLAM (PTAM) to fit a dense surface estimate of the environment that is refined using direct methods. A surface-based model is then computed and polygonized to best fit the triangulated landmarks from the feature-based front end. A parallel process chooses a batch of frames that have a potentially overlapping surface visibility, in order to estimate a dense refinement over the base mesh, using a GPU accelerated implementation of variational optical flow.

In an update to this work, Newcombe published, in 2011, Dense Tracking and Mapping in Real-Time (DTAM) [56] that removed the need for PTAM as a front-end to the system, and generalized the dense reconstruction to fully solve the monocular SLAM pipeline, by performing, online, a dense reconstruction, given camera pose estimates that are found through whole image alignment.

Similar to the work of Newcombe [56], [57] modeled the environment as a 3D piecewise smooth surface, and used a sparse feature based front-end as a base for a Delaunay triangulation to fit a mesh that is used to interpolate a dense reconstruction of the environment.

[58] released CD SLAM in 2011, with the objectives to handle short- and long-term environmental changes and to handle mixed indoor/outdoor environments. To limit the map size and gain robustness against significant rotational changes, CD SLAM suggests the use of a modified Histogram of Oriented Cameras descriptor (HOC) [84], with a GPU accelerated descriptor update, and a probabilistic weighting scheme to handle outliers. Furthermore, it suggests the use of large-scale nested loop closures with scale drift correction, and provide a geometric adaptation to update the feature descriptors after loop closure. Keyframes are organized in an undirected, unweighted pose graph. Re-localization is performed using a non-linear least square minimization, initialized with the

pose of the best matching candidate keyframe from the map, found through FABMAP.

RD SLAM [60] was released with the aim to handle occlusions and slowly varying, dynamic scenes. RD SLAM employs a heavily parallelized GPU accelerated SIFT and stores them in a KD-Tree [85] that further accelerates feature matching based on the nearest neighbor of the queried feature in the tree. While the KD-tree is meant to accelerate SIFT feature matching, updating it with new features is computationally intensive. RD SLAM suggests the usage of another tree, alternating between both, activating one tree at a time for matching, and another tree is passively waiting to be updated when a sufficient number of new SIFT features is observed. RD SLAM disregards feature matches that exhibit a viewpoint angle difference larger than thirty degrees. To cope with dynamic objects and slowly varying scenes, RD SLAM suggests a prior-based adaptive RANSAC scheme that samples, based on the outlier ratio of features in previous frames, the features in the current frame from which to estimate the camera pose. Furthermore, it performs landmark and keyframe culling, using histograms of colors to detect and update changed image locations, while sparing temporarily occluded landmarks.

[61] dealt with the problem of pure rotations in the camera motion by building local panorama maps, whenever the system explores a new scene with pure rotational motion. The system extracts phonySIFT descriptors as described in [86] and establishes feature correspondences using an accelerated matching method through hierarchical k-means. When insufficient 3D landmarks are observed during pose estimation, the system transitions into a rotation-only estimation mode and starts building a panorama map until the camera observes part of the finite map.

Also in 2013, [87] published MonoFusion: Real-time 3D Reconstruction of Small Scenes with a Single Web Camera. MonoFusion estimates a dense depth map for every tracked frame I by selecting, from a pool of its surrounding keyframes, the keyframe I' that shares with it the most number of feature matches. MonoFusion then proceeds by assigning for every pixel a set of random depth values, and chooses the depth measurement that yield the least reprojection error between the two frames. Post processing steps are then employed to remove depth measurements with high reprojection error and isolated regions of similar depths (as they are most likely to be outliers). Finally a volumetric volume of the reconstructed scene is obtained by fusing the depth maps of every frame into a signed distance field (SDF), as described in [88].

In the work of [40], the sought after objective is to handle tracking, mapping, and loop closure, all using the same binary feature, through a hybrid map representation. Whenever a loop is detected, the map is converted to its metric form, where a local bundle adjustment take place before returning the map back to its topological form.

[65] published an offline monocular SLAM system, which employs a divide- and-conquer strategy, by segmenting the map into submaps. A similarity transform is estimated between each submap and its ten nearest neighbors. A global similarity transform, relating every submap to a single global reference frame, is computed by a pose graph optimization, where the reference frames are stored in a graph of submaps. The above procedure is susceptible to outliers in the loop detection module, and hence the need for an efficient outlier handling mechanism. For such purpose, temporally consecutive similarity measurements are always considered as inliers. The outlier rejection module proceeds then by integrating the similarities over the shortest loop it can

find, and monitors the closure error to accept the loop. To cope with a very large number of submaps, a loopy belief propagation algorithm cuts the main graph into subgraphs, before a non-linear optimization takes place.

In 2016, [67] published Multi-Level Mapping: Real-time Dense Monocular SLAM, which is built on top of LSD SLAM, and allows it to generate and track a dense map, in real-time; in contrast to LSD SLAM's default semi-dense map. The key contribution that allowed such real-time operation is the introduction of Quad-trees keyframe representations: a pyramid level representation of the keyframe is generated, and stored in trees. The depth and variance measurements, estimated in a similar fashion to LSD SLAM, at the higher pyramid levels of the keyframes, allow for a denser representation of low-texture regions, once projected back onto the full image resolution. Holes, corresponding to pixels that failed to converge during their depth estimates are then filled using its surrounding neighboring pixels at their appropriate pyramid levels before being mapped back onto the full resolution. Spatial regularization is then employed to remove outlier measurements, and smooth the noise in the estimated depth map. Finally, the estimated dense map undergoes a pose graph optimization over $sim(3)$ that ensures its alignment with surrounding keyframes, and allows its incorporation within the global map.

In 2016, **RKSLAM** (Robust keyframe based monocular SLAM for augmented reality) [68] was published with the goal to robustify monocular slam algorithms for the scenarios encountered during augmented reality sessions, specifically when the camera undergoes fast motions performed by untrained personnel. RKSLAM argues that for erratic motions, motion models as employed by other systems leads to drastic failure, and suggests to employ three different Homographies to solve for the SLAM task:

1. A global Homography that relates the current frame to other keyframes in the map,
2. A set of local Homographies extracted between the current frame and another keyframe through RANSAC and finally
3. Planar specific Homographies relating specific planar surfaces observed in a keyframe and the current frame.

The Homographies are used to estimate a set of 3D points possibly available in the current field of view, and accordingly warp them taking into account the viewpoint changes from which they were initially observed, and finally project them on the current image. To allow the system to accommodate much needed features as the user performs fast maneuvers, the local mapping thread was moved from the backend of the system to its frontend and operates simultaneously with the tracking thread, at frame-rate. The local mapping system is responsible for maintaining and generating weakly supported 3D landmarks that have not yet been fully optimized by the global mapping thread that runs in the background.

Most of the aforementioned closed source systems are not as popular as their open source counterparts; however, they influenced the development of their open-source counterparts. They also provide a lot of constructive ideas that can be built upon to enhance the open source systems. For example ideas put forward by RDSLAM can be adapted to robustify KSLAM systems in dynamic and occluded environments. Other ideas such as the adoption of a photogenerative model in monocular SLAM by [53] was recently employed in the open source work of [69].

5. Discussion

During the preparation of this survey, we considered including a benchmark, in which the different monocular SLAM systems in the literature are compared. However, each different flavor of monocular SLAM is favored by different operational conditions.

For example, SVO [63] prefers high frame rate inputs from downward looking cameras, DPPTAM [66] can only operate in indoor environments where most of the observed scene is composed of planar surfaces. DT SLAM [64] requires the scene to be repeatedly observed. Accordingly, there does not exist any public dataset in the literature that would allow us to perform an unbiased experimental comparison across all systems. Instead, in this section, we discuss and evaluate the ramifications of the decisions made in each component of the different monocular SLAM systems, providing a theoretical insight into the limitations of the various modules designs.

5.1. Traits of data association

5.1.1. Traits of direct methods
Direct methods exploit all information available in the image and are therefore more robust than feature-based methods in regions with poor texture and blur. Nevertheless, direct methods are susceptible to scene illumination changes, due to the violation of the underlying brightness consistency assumption (Eq. (1)) . In an effort to gain resilience against this mode of failure, the recently released DSO models the image formation process, and attempts to incorporate the scene irradiance in the energy functional, at the expense of adding a calibrated image formation model which is used to correct the images at a pre-processing step. The model is estimated through an additional offline calibration process described in [89].

Furthermore, during the non-linear optimization process, Eq. (2), is linearized through a first order Taylor expansion. While the linearizion is valid when the parameters of the warping transform tends to zero, higher order terms becomes dominant and the linearizion becomes invalid for large transforms. Therefore, a second disadvantage of direct methods is the assumption of small motions between the images (typically not more than 1 pixel). To relax this constraint, direct monocular SLAM systems employ a pyramidal implementation, where the image alignment process takes place sequentially from the highest pyramid level to the lowest, using the results of every level as a prior to the next level. They also suggest the usage of high frame rate cameras to alleviate this issue; some systems employ an efficient second order minimization (ESM [75]) to estimate a rotation prior that helps increase the convergence radius. Despite these efforts, the tolerated baseline for data association in direct methods is considerably smaller than the tolerated baseline in feature-based methods.

Another disadvantage of direct methods is that the calculation of the photometric error at every pixel is computationally intensive; therefore, real-time monocular SLAM applications of direct methods, until recently, were not considered feasible. However, with the recent advancements in parallelized processing and with the introduction of semi-dense inverse depth filtering, it became possible to integrate direct methods into KSLAM solutions [63,66,90].

5.1.2. Traits of feature-based methods
Feature-based methods are relatively robust to lighting changes and can tolerate wider baselines; however, the extraction processes that make them resilient to these factors are generally computationally expensive. For real-time operation constraints, most systems employ a trade-of between a feature type to use in one hand, and the robustness and resilience to environment factors on the other. To mitigate this constraint, other systems, such as the work of [60], resort to parallelized GPU implementations for feature detection and extraction.

Another disadvantage of feature-based methods is that even the top performing feature descriptors are limited in the amount of scene change (lighting and viewpoint) they can handle before
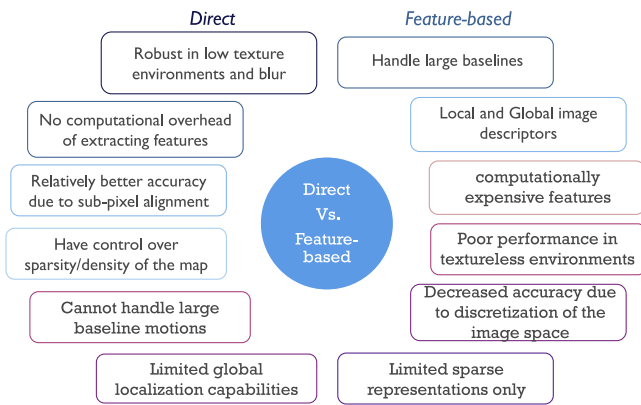
Fig. 16. Comparison between the different traits of direct vs. feature-based KSLAM systems.

failure. Feature matching is also prone to failure in similar-self-repeating texture environments, where a feature in I_1 can be ambiguously matched to multiple other features in I_2. Outliers in the data association module can heavily degrade the system performance by inducing errors in both the camera poses and the generated map until the point of failure. Feature-based methods also suffer from lack of features in textureless regions, causing feature-based KSLAM to fail in texture-deprived environments. A summary of the comparison between direct and feature-based methods is shown in Fig. 16.

5.1.3. Traits of the different data association types

In general establishing data associations remains one of the biggest challenges in KSLAM. Systems that limit the search range along the epipolar line using the observed depth information, implicitly assume a relatively smooth depth distribution. Violating this assumption (*i.e.*, when the scene includes significance variance in the observed depth) causes the 2D features corresponding to potential future 3D landmarks to fall outside the boundaries of the epipolar segment, and the system ends up neglecting them. Other limitations for data association arise from large erratic accelerations in the camera's motion, also causing features to fall outside the scope of the search window. Such a scenario is common when the camera is operated by an untrained user. Under the same type of motions, image pollution with motion blur also negatively impacts the performance of data association methods to the point of failure.

Erroneous data association is also a very common problem that can cause false positives in self repeating environments. Most current implementations of data association address this problem through a bottom-up approach, where low level information from image pixels or from features, is used to establish correspondences. To mitigate some of these issues, a number of systems have attempted to use geometric features of a higher significance, such as lines [91–98], superpixels or planar features [66,99,100], or priors on 3D shapes in the scene [101]. Recent advances in machine learning are promising alternatives to remedy some of the data association issues by automatically learning to extract and match features using the methods suggested and not limited to [102–106].

5.2. Traits of initialization

Aside from the random depth initialization of LSD SLAM and DSO, all the suggested methods described above suffer from degeneracies under certain conditions, such as under low-parallax movements of the camera, or when the scene's structure assumption – the

Fundamental matrix's assumption for general non-planar scenes or the Homography's assumption of planar scenes – is violated.

PTAM's initialization procedure is brittle and remains tricky to perform, especially for inexperienced users. Furthermore, it is subject to degeneracies when the planarity of the initial scene's assumption is violated, or when the user's motion is inappropriate; thereby crashing the system, without means of detecting such degeneracies.

As is the case in PTAM, the initialization of SVO requires the same type of motion and is prone to sudden movements, as well as to non-planar scenes. Furthermore, monitoring the median of the baseline distance between features is not a good approach to automate the initial keyframe pair selection, as it is prone to failure against degenerate cases, with no means of detecting them.

The model based initialization of ORB SLAM attempts to automatically initialize the system by monitoring the baseline and the scene across a window of images. If the observed scene is relatively far, while the camera slowly translates in the scene, the system is not capable of detecting such scenarios, and fails to initialize.

While a random depth initialization from a single image does not suffer from the degeneracies of two view geometry methods, the depth estimation requires the processing of subsequent frames to converge, resulting in an intermediate tracking phase where the generated map is not reliable. It requires slow translational motions, and the convergence of the initialization is not guaranteed.

5.3. Traits of pose estimation

Systems relying on constant motion models, such as PTAM and ORB SLAM are prone to tracking failure when abrupt changes in the direction of the camera's motion occurs. While they both employ a recovery from such failures, PTAM's tracking performance is exposed to false positive pose recovery; as opposed to ORB SLAM that first attempts to increase the search window before invoking its failure recovery module.

Another limitation of feature-based pose estimation is the detection and handling of occlusions. As the camera translates in the scene, some landmarks in the background are prone to occlusions from objects in the foreground. When the system projects the 3D map points onto the current frame, it fails to match the occluded features, and counts them toward the camera tracking quality assessment. In extreme cases, the tracking quality of the system might be deemed bad and tracking failure recovery procedures are invoked even though camera pose tracking did not fail. Furthermore, occluded points are flagged as outliers and passed to the map maintenance module to be removed, depriving the map from valid useful landmarks that were erroneously flagged due to occlusions in the scene.

Other systems, that use the previously tracked pose as a prior for the new frame's pose, are also prone to the same limitations of constant velocity models. Furthermore, they require small displacements between frames, limiting their operation to relatively expensive high frame rate cameras (typically > 70 *fps*) such that the displacement limitation is not exceeded. Another limitation of these methods is inherent from their use of direct data association. Their tracking module is susceptible to variations in the lighting conditions. To gain some resilience to lighting changes in direct methods, [69] suggest an off-line photometric calibration process to parametrize and incorporate lighting variations within the camera pose optimization process.

A common limitation that plagues most tracking modules is the presence of dynamic objects in the observed environment. As most KSLAM systems assume a static scene, the tracking modules of most systems suffer from tracking failures: a significantly large dynamic object in the scene could trick the system into thinking that the camera itself is moving, while it did not move relative to

the environment. Small, slowly moving objects can introduce noisy outlier landmarks in the map and require subsequent processing and handling to be removed. Small, fast moving objects on the other hand, do not affect the tracking module as much. 2D features corresponding to fast moving small objects tend to violate the epipolar geometry of the pose estimation problem, and are easily flagged and removed from the camera pose optimization thread; however, they can occlude other landmarks. To address the effects of occlusions and dynamic objects in the scene, [60] suggest, for slowly varying scenes, to sample based on previous camera pose locations in the image that are not reliable, and discard them during the tracking phase.

5.4. Traits of map generation

A major limitation in the triangulation-by-optimization method is the requirement of a significant baseline separating two viewpoints observing the same feature. Hence, it is prone to failure when the camera's motion is made of pure rotations. To counter such modes of failure, DT SLAM introduced 2D landmarks that can be used to expand the map during pure rotations, before they are triangulated into 3D landmarks. However, the observed scene during the rotation motion is expected to be re-observed with more baseline, for the landmarks to transition from 2D to 3D. Unfortunately, in many applications this is not the case; for example, a camera mounted on a car making a turn cannot re-observe the scene, and eventually tracking failure occurs. DT SLAM addresses such cases by generating a new sub map and attempts to establish connections to previously created sub-maps by invoking a thread to look for similar keyframes across sub-maps, and establish data associations between them. In the mean time, it resumes tracking in the new world coordinate frame of the new sub-map. This however renders the pose estimates obsolete; at every tracking failure, the tracking is reset to the new coordinate frame, yielding useless pose estimates until the sub-maps are joined together, which may never occur.

In filter based triangulation methods, outliers are easily flagged as landmarks whose distribution remain approximately uniform after a number of observations have been incorporated in the framework. This reduces the need for a subsequent processing step to detect and handle outliers. Also, landmarks at infinity feature parallax values that are too small for triangulation purposes; but yet, can be used to enhance the camera's rotation estimates, and kept in the map, and are transitioned from infinity to the metric map, when enough parallax between the views observing them is recorded. However, these benefits come at the expense of increased complexity in implementing a probabilistic framework, which keeps track and updates the uncertainty in the depth distribution of every pixel with a gradient in the system.

Furthermore, while the dense and semi-dense maps can capture a much more meaningful representation of a scene than a sparse set of 3D landmarks, the added value is diminished by the challenges of handling immense amounts of data in 3D. Hence there is a need for additional, higher level semantic information to reason about the observed scene, and to enhance the system's overall performance. While monocular SLAM systems have been shown to improve the results of semantic labeling [107], the feedback from the latter to the former remains a challenging problem. Previous work on the matter include but are not limited to [101,108–111].

5.5. Traits of BA/PGO/map maintenance

Pose Graph Optimization (PGO) returns inferior results to those produced by GBA, while PGO optimizes only for the keyframe poses – and accordingly adjusts the 3D structure of landmarks – GBA and LBA jointly optimize for both keyframe poses and 3D structure. The stated advantage comes at the cost of computational time, with PGO exhibiting significant speed up compared to the other methods. PGO is often employed during loop closure as the computational cost of running a full BA is often intractable on large-scale loops; however, pose graph optimization may not yield optimal result if the errors accumulated over the loop are distributed along the entire map, leading to locally induced inaccuracies in regions that were not originally wrong.

5.6. Traits of global localization

For successful re-localization or loop detection, global localization methods employed by PTAM, SVO and DT SLAM require the camera's pose to be near the recorded keyframe's pose, and would otherwise fail when there is a large displacement between the two. Furthermore, they are highly sensitive to any change in the lighting conditions of the scene, and may yield many false positives when the observed environment is composed of self repeating textures. Other methods that rely on bags of words representation of high dimensional features are susceptible to failure when the training set of the bag of words classifier is not representative of the working environment in which the system is operating.

6. Conclusions

During the course of our review, we have outlined the essential building blocks of a generic monocular SLAM system; including data association, visual initialization, pose estimation, topological/metric map generation, BA/PGO/map maintenance, and global localization. We have also discussed the details of the latest open-source state of the art systems in monocular SLAM including PTAM, SVO, DT SLAM, LSD SLAM, ORB SLAM, DPPTAM, and DSO. Finally, we compiled and summarized what added information closed-source keyframe-based monocular SLAM systems have to offer. Although extensive research has been dedicated to this field, it is our opinion that each of the building blocks discussed above could benefit from many improvements of which we list the following:

- Robust data association against illumination changes, dynamic scenes, and occluded environments,
- A robust initialization method that can operate without an initial scene assumption,
- An accurate camera pose estimate that is not affected by sudden movements, blur, noise, large depth variations, nor moving objects,
- A map making module capable of generating an efficient dense scene representation in regions of little texture, while incorporating a higher level of perception,
- A map maintenance method that improves the map, with resilience against dynamic, changing environments, and finally,
- A failure recovery procedure capable of reviving the system from significantly large changes in camera viewpoints.

These are all desired properties that remain challenging topics in the field of keyframe-based monocular Visual SLAM. Furthermore, with the recent advancements in machine learning, researchers are moving toward integrating semantic data within the context of VSLAM. While the incorporation of semantic data into VSLAM is undoubtedly the next step in the right direction, we argue that such integration requires a hybrid fusion approach that tightly integrates metric, topological and semantic representations in a symbiotic relationship, a research area relatively uncharted.

References

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, J. Leonard, Past, present, and future of simultaneous localization and mapping: toward the robust-perception age, IEEE Trans. Robot. 32 (2016) 1309–1332.

[2] D. Scaramuzza, F. Fraundorfer, Visual odometry [tutorial], IEEE Robot. Autom. Mag. 18 (2011) 80–92. http://dx.doi.org/10.1109/MRA.2011.943233.

[3] J. Fuentes-Pacheco, J. Ruiz-Ascencio, J.M. Rendón-Mancha, Visual simultaneous localization and mapping: A survey, Artif. Intell. Rev. 43 (2012) 55–81. http://dx.doi.org/10.1007/s10462-012-9365-8.

[4] K. Yousif, A. Bab-Hadiashar, R. Hoseinnezhad, An overview to visual odometry and visual SLAM: Applications to mobile robotics, Intell. Ind. Syst. 1 (2015) 289–311.

[5] S. Saeedi, M. Trentini, M. Seto, H. Li, Multiple-Robot simultaneous localization and mapping: A Review, J. Field Robot. 33 (2016) 3–46. http://dx.doi.org/10.1002/rob.21620.

[6] T. Bailey, H. Durrant-Whyte, Simultaneous localization and mapping (SLAM): Part II, IEEE Robot. Autom. Mag. 13 (2006) 108–117. http://dx.doi.org/10.1109/MRA.2006.1678144.

[7] B.D. Lucas, T. Kanade, An Iterative Image Registration Technique with an Application to Stereo Vision, in: International Joint Conference on Artificial Intelligence - vol. 2, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1981, pp. 674–679.

[8] S. Baker, I. Matthews, Lucas-Kanade 20 years on: A unifying framework, Int. J. Comput. Vis. 56 (2004) 221–255.

[9] S. Krig, Interest point detector and feature descriptor survey, in: Computer Vision Metrics: Survey, Taxonomy, and Analysis, Apress, Berkeley, CA, 2014, pp. 217–282. http://dx.doi.org/10.1007/978-1-4302-5930-5.6.

[10] P.R. Beaudet, Rotationally invariant image operators, in: International Conference on Pattern Recognition, 1978.

[11] C. Harris, M. Stephens, A combined corner and edge detector, In: Proc. of Fourth Alvey Vision Conference, pp. 147–151, 1988.

[12] J. Shi, C. Tomasi, Good features to track, in: Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94, 1994 IEEE Computer Society Conference on, 1994, pp. 593–600.

[13] T. Lindeberg, Feature detection with automatic scale selection, Int. J. Comput. Vis. 30 (1998) 79–116. http://dx.doi.org/10.1023/A:1008045108935.

[14] J. Matas, O. Chum, M. Urban, T. Pajdla, Robust wide baseline stereo from maximally stable extremal regions, in: Proc. BMVC, 36.1–36.10. http://dx.doi.org/10.5244/C.16.36.

[15] D.G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vis. 60 (2004) 91–110. http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94.

[16] E. Mair, G.D. Hager, D. Burschka, M. Suppa, G. Hirzinger, Adaptive and generic corner detection based on the accelerated segment test, in: Proceedings of the European Conference on Computer Vision, ECCV'10, 2010. http://dx.doi.org/10.1007/978-3-642-15552-9_14.

[17] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, P. Fua, Brief: computing a local binary descriptor very fast, IEEE Trans. Pattern Anal. Mach. Intell. 34 (2012) 1281–1298. http://dx.doi.org/10.1109/TPAMI.2011.222.

[18] S. Leutenegger, M. Chli, R.Y. Siegwart, Brisk: Binary robust invariant scalable keypoints, Computer Vision, ICCV, 2011 IEEE International Conference on, 2011, pp. 2548–2555. http://dx.doi.org/10.1109/ICCV.2011.6126542.

[19] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, Speeded-up robust features (SURF), Comput. Vis. Image Underst. 110 (2008) 346–359. http://dx.doi.org/10.1016/j.cviu.2007.09.014.

[20] D. Lowe, Object recognition from local scale-invariant features, in: International Conference on Computer Vision, ICCV, the Seventh IEEE, vol. 2, IEEE, 1999, pp. 1150–1157.

[21] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection,Computer Vision and Pattern Recognition, in: 2005. CVPR 2005. IEEE Computer Society Conference on, vol. 1, 2005, pp. 886–893. http://dx.doi.org/10.1109/CVPR.2005.177.

[22] A. Alahi, R. Ortiz, P. Vandergheynst, Freak: Fast retina keypoint, in: Computer Vision and Pattern Recognition, CVPR, 2012 IEEE Conference on, 2012, pp. 510–517. http://dx.doi.org/10.1109/CVPR.2012.6247715.

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

[24] P. Moreels, P. Perona, Evaluation of features detectors and descriptors based on 3D objects, Int. J. Comput. Vis. 73 (2007) 263–284. http://dx.doi.org/10.1007/s11263-006-9967-1.

[25] J. Hartmann, J.H. Klussendorff, E. Maehle, A comparison of feature descriptors for visual SLAM, in: Mobile Robots, ECMR, 2013 European Conference on, 2013, pp. 56–61. http://dx.doi.org/10.1109/ECMR.2013.6698820.

[26] I. Rey-Otero, M. Delbracio, J. Morel, Comparing feature detectors: A bias in the repeatability criteria, and how to correct it, CoRR (2014) abs/1409.2465. http://arxiv.org/abs/1409.2465.

[27] A. Hietanen, J. Lankinen, J.-K. Kämäräinen, A.G. Buch, N. Krüger, A comparison of feature detectors and descriptors for object class matching, Neurocomputing (2016).

[28] J.L.C. Jérôme Martin, Experimental comparison of correlation techniques, in: IAS-4, International Conference on Intelligent Autonomous Systems, 1995.

[29] M. Muja, D.G. Lowe, Fast approximate nearest neighbors with automatic algorithm configuration, in: VISAPP International Conference on Computer Vision Theory and Applications, 2009, pp. 331–340.

[30] D. Galvez-López, J.D. Tardos, Bags of binary words for fast place recognition in image sequences, IEEE Trans. Robot. 28 (2012) 1188–1197. http://dx.doi.org/10.1109/TRO.2012.2197158.

[31] S. Umeyama, Least-squares estimation of transformation parameters between two point patterns, IEEE Trans. Pattern Anal. Mach. Intell. 13 (1991) 376–380. http://dx.doi.org/10.1109/34.88573. http://dx.doi.org/10.1109/34.88573.

[32] A.J. Davison, I.D. Reid, N.D. Molton, O. Stasse, MonoSLAM: Real-time single camera SLAM, IEEE Trans. Pattern Anal. Mach. Intell. 29 (2007) 1052–1067.

[33] H.C. Longuet-Higgins, A computer algorithm for reconstructing a scene from two projections, Lett. Nature 293 (1981) 133–135. http://dx.doi.org/10.1038/293133a0.

[34] P.H.S. Torr, A. Zisserman, MLESAC, Comput. Vis. Image Underst. 78 (2000) 138–156.

[35] R. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge University Press, 2003, p. 655.

[36] J. Boal, Á. Sánchez-Miralles, Á. Arranz, Topological simultaneous localization and mapping: A survey, Robotica 32 (2014) 803–821.

[37] R. Mur-Artal, J.M.M. Montiel, J.D. Tardos, ORB-SLAM: A Versatile and Accurate Monocular SLAM System, IEEE Trans. Robot. PP (2015) 1–17.

[38] J. Engel, T. Schöps, D. Cremers, LSD-SLAM: Large-Scale Direct Monocular SLAM, in: D. Fleet, T. Pajdla, B. Schiele, T. Tuytelaars (Eds.), Computer Vision – ECCV 2014 SE - 54, in: Lecture Notes in Computer Science, vol. 8690, Springer International Publishing, 2014, pp. 834–849.

[39] J. Lim, J.M. Frahm, M. Pollefeys, Online environment mapping, in: Computer Vision and Pattern Recognition, CVPR, 2011 IEEE Conference on, pp. 3489–3496, 2011.

[40] H. Lim, J. Lim, H.J. Kim, Real-time 6-DOF monocular visual SLAM in a large-scale environment, in: Robotics and Automation, ICRA, IEEE International Conference on, 2014, pp. 1532–1539.

[41] E. Fernández-Moral, V. Arévalo, J. González-Jiménez, Hybrid Metric-topological Mapping for Large Scale Monocular SLAM, Springer International Publishing, 2015, pp. 217–232.

[42] K. Konolige, Sparse sparse bundle adjustment, in: Proceedings of the British Machine Vision Conference, BMVA Press, 2010, pp. 102.1–102.11. http://dx.doi.org/10.5244/C.24.102.

[43] R.I. Hartley, P. Sturm, Triangulation, Comput. Vis. Image Underst. 68 (1997) 146–157. http://dx.doi.org/10.1006/cviu.1997.0547.

[44] S. Hochdorfer, C. Schlegel, Towards a robust visual SLAM approach: Addressing the challenge of life-long operation, in: Advanced Robotics, 2009. ICAR 2009. International Conference on, 2009, pp. 1–6.

[45] J. Civera, A. Davison, J. MontielMontiel, Inverse depth parametrization for monocular SLAM, IEEE Trans. Robot. 24 (2008) 932–945.

[46] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, W. Burgard, G2o: A general framework for graph optimization, in: Robotics and Automation, ICRA, IEEE International Conference on, IEEE, 2011, pp. 3607–3613.

[47] B. Triggs, P.F. McLauchlan, R.I. Hartley, A.W. Fitzgibbon, Springer Berlin Heidelberg, Berlin, Heidelberg, 2000, pp. 298–372. http://dx.doi.org/10.1007/3-540-44480-7.21.

[48] H. Strasdat, A.J. Davison, J.M.M. Montiel, K. Konolige, Double Window Optimisation for Constant Time Visual SLAM, in: International Conference on Computer Vision, Proceedings of the, ICCV '11, IEEE Computer Society, Washington, DC, USA, 2011, pp. 2352–2359.

[49] E. Garcia-Fidalgo, A. Ortiz, Vision-based topological mapping and localization methods: a survey, Robot. Auton. Syst. 64 (2015) 1–20. http://dx.doi.org/10.1016/j.robot.2014.11.009. http://www.sciencedirect.com/science/article/pii/S0921889014002619.

[50] S. Agarwal, K. Mierle, et al., Ceres solver, 2013.

[51] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, P. Sayd, Real time localization and 3D reconstruction, in: computer vision and pattern recognition, 2006 IEEE Computer Society Conference on, vol. 1, 2006. pp. 363–370. http://dx.doi.org/10.1109/CVPR.2006.236.

[52] G. Klein, D. Murray, parallel tracking and mapping for small AR workspaces, in: 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, 2007, pp. 1–10.

[53] G. Silveira, E. Malis, P. Rives, An efficient direct approach to visual SLAM, IEEE Trans. Robot. 24 (2008) 969–979. http://dx.doi.org/10.1109/TRO.2008.2004829.

[54] H. Strasdat, J. Montiel, A. Davison, Scale Drift-Aware Large Scale Monocular SLAM, The MIT Press, 2010. http://www.roboticsproceedings.org/rss06/.

[55] R.A. Newcombe, A.J. Davison, Live dense reconstruction with a single moving camera, in: Computer Vision and Pattern Recognition, CVPR, 2010 IEEE Conference on, IEEE, 2010, pp. 1498–1505.

[56] R.A. Newcombe, S.J. Lovegrove, A.J. Davison, DTAM: dense tracking and mapping in real-time, in: Proceedings of the 2011 International Conference on Computer Vision, IEEE Computer Society, Washington, DC, USA, 2011, pp. 2320–2327. http://dx.doi.org/10.1109/ICCV.2011.6126513.

[57] A. Pretto, E. Menegatti, E. Pagello, Omnidirectional dense large-scale mapping and navigation based on meaningful triangulation, in: Robotics and Automation, ICRA, 2011 IEEE International Conference on, 2011, pp. 3289–3296. http://dx.doi.org/10.1109/ICRA.2011.5980206.

[58] K. Pirker, M. Rüther, H. Bischof, CD SLAM - Continuous localization and mapping in a dynamic world., in: IEEE International Conference on Intelligent Robots Systems, IROS, IEEE, 2011, pp. 3990–3997.

[59] C. Pirchheim, G. Reitmayr, (2011) Homography-based planar mapping and tracking for mobile phones, in: Mixed and Augmented Reality, ISMAR, 2011 10th IEEE International Symposium on, 2011, pp. 7–36. http://dx.doi.org/10.1109/ISMAR.2011.6092367.

[60] W. Tan, H. Liu, Z. Dong, G. Zhang, H. Bao, Robust monocular SLAM in dynamic environments, in: 2013 IEEE International Symposium on Mixed and Augmented Reality, ISMAR, 2013, pp. 209–218.

[61] C. Pirchheim, D. Schmalstieg, G. Reitmayr, Handling pure camera rotation in keyframe-based SLAM, in: Mixed and Augmented Reality, ISMAR, 2013 IEEE International Symposium on, 2013, pp. 229–238. http://dx.doi.org/10.1109/ISMAR.2013.6671783.

[62] Z. Dong, G. Zhang, J. Jia, H. Bao, Efficient keyframe-based real-time camera tracking, Comput. Vis. Image Underst. 118 (2014) 97–110. http://dx.doi.org/10.1016/j.cviu.2013.08.005. http://www.sciencedirect.com/science/article/pii/S1077314213001562.

[63] C. Forster, M. Pizzoli, D. Scaramuzza, SVO : Fast semi-direct monocular visual odometry, in: Robotics and Automation, ICRA, IEEE International Conference on, 2014.

[64] D. Herrera, J. Kannala, K. Pulli, J. Heikkila, DT-SLAM: Deferred triangulation for robust SLAM, in: 3D Vision, 2nd International Conference on, vol. 1, IEEE, 2014, pp. 609–616.

[65] G. Bourmaud, R. Megret, Robust large scale monocular visual SLAM, in: Computer Vision and Pattern Recognition, CVPR, 2015 IEEE Conference on, 2015 pp. 1638–1647. http://dx.doi.org/10.1109/CVPR.2015.7298772.

[66] A. Concha, J. Civera, DPPTAM: Dense piecewise planar tracking and mapping from a monocular sequence, in: Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, 2015, pp. 5686–5693. http://dx.doi.org/10.1109/IROS.2015.7354184.

[67] W.N. Greene, K. Ok, P. Lommel, N. Roy, (2016) Multi-level mapping: Real-time dense monocular SLAM, 2016, pp. 833–840. http://dx.doi.org/10.1109/ICRA.2016.7487213.

[68] H. Liu, G. Zhang, H. Bao, Robust keyframe-based monocular SLAM for augmented reality, in: International Symposium on Mixed and Augmented Reality, ISMAR, 2016.

[69] J. Engel, V. Koltun, D. Cremers, Direct sparse odometry, CoRR abs/1607.02565 (2016). http://arxiv.org/abs/1607.02565.

[70] E. Rosten, T. Drummond, Machine Learning for High-speed Corner Detection, in: 9th European Conference on Computer Vision - Volume Part I, Proceedings of the, Springer-Verlag, Berlin, Heidelberg, 2006, pp. 430–443.

[71] D. Nistér, An efficient solution to the five-point relative pose problem, IEEE Trans. Pattern Anal. Mach. Intell. 26 (2004) 756–777.

[72] O. Faugeras, F. Lustman, Motion and structure from motion in a piecewise planar environment, Int. J. Pattern Recognit. Artif. Intell. 02 (1988).

[73] C. Tomasi, T. Kanade, Detection and Tracking of Point Features, 1991.

[74] B.C. Hall, Lie Groups, Lie Algebras, and Representations, vol. 222, Number 102 ed., Springer- Verlag, 2015.

[75] S. Benhimane, E. Malis, Homography-based 2D Visual Tracking and Servoing, Int. J. Robot. Res. 26 (2007) 661–676.

[76] R.A. Moranna, R.D. Martin, V.J. Yohai, Robust Statistics, Wiley, 2006.

[77] L. Kneip, R. Siegwart, M. Pollefeys, Finding the Exact Rotation between Two Images Independently of the Translation, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 696–709. http://dx.doi.org/10.1007/978-3-642-33783-3_50. Chapter Finding th..

[78] J. Engel, J. Sturm, D. Cremers, Semi-dense Visual Odometry for a Monocular Camera, in: Computer Vision, ICCV, IEEE International Conference on, IEEE, 2013, pp. 1449–1456.

[79] G. Vogiatzis, C. Hernández, Video-based, real-time multi-view stereo, Image Vis. Comput. 29 (2011) 434–441. http://dx.doi.org/10.1016/j.imavis.2011.01.006.

[80] M. Pizzoli, C. Forster, D. Scaramuzza, REMODE: Probabilistic, monocular dense reconstruction in real time, in: IEEE International Conference on Robotics and Automation, ICRA, 2014.

[81] V. Lepetit, F. Moreno-Noguer, P. Fua, EPnP: An Accurate O(n) Solution to the PnP Problem, Int. J. Comput. Vis. 81 (2009) 155–166.

[82] A. Glover, W. Maddern, M. Warren, S. Reid, M. Milford, G. Wyeth, OpenFABMAP: An open source toolbox for appearance-based loop closure detection, in: 2012 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2012, pp. 4730–4735.

[83] Z. Dong, G. Zhang, J. Jia, H. Bao, Keyframe-based real-time camera tracking, in: 2009 IEEE 12th International Conference on Computer Vision, 2009, pp. 1538–1545. http://dx.doi.org/10.1109/ICCV.2009.5459273.

[84] K. Pirker, Histogram of oriented cameras - a new descriptor for visual SLAM in dynamic environments, in: Proceedings of the British Machine Vision Conference, BMVA Press, 2010, pp. 76.1–76.12. http://dx.doi.org/10.5244/C.24.76.

[85] J.L. Bentley, Multidimensional Binary Search Trees Used for Associative Searching, Commun. ACM 18 (1975) 509–517.

[86] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, D. Schmalstieg, Real-time detection and tracking for augmented reality on mobile phones, IEEE Trans. Vis. Comput. Graphics 16 (2010) 355–368. http://dx.doi.org/10.1109/TVCG.2009.99.

[87] V. Pradeep, C. Rhemann, S. Izadi, C. Zach, M. Bleyer, S. Bathiche, Monofusion: real-time 3D reconstruction of small scenes with a single web camera, in: Mixed and Augmented Reality, ISMAR, 2013 IEEE International Symposium on, IEEE, 2013, pp. 83–88.

[88] B. Curless, M. Levoy, A volumetric method for building complex models from range images, in: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, ACM, New York, NY, USA, 1996, pp. 303–312. http://dx.doi.org/10.1145/237170.237269.

[89] J. Engel, V.C. Usenko, D. Cremers, A photometrically calibrated benchmark for monocular visual odometry, CoRR (2016) abs/1607.02555. http://arxiv.org/abs/1607.02555.

[90] J. Engel, J. Stuckler, D. Cremers, Large-scale direct SLAM with stereo cameras, Intelligent Robots and Systems, IROS, in: 2015 IEEE/RSJ International Conference on,2015, pp. 1935–1942. http://dx.doi.org/10.1109/IROS.2015.7353631.

[91] S.R. Bista, P.R. Giordano, F. Chaumette, Appearance-Based indoor navigation by ibvs using line segments, IEEE Robot. Autom. Lett. 1 (2016) 423–430. http://dx.doi.org/10.1109/LRA.2016.2521907.

[92] L. Zhang, R. Koch, Hand-Held monocular SLAM based on line segments, in: Proceedings of the 2011 Irish Machine Vision and Image Processing Conference, IMVIP '11, IEEE Computer Society, Washington, DC, USA, 2011, pp. 7–14. http://dx.doi.org/10.1109/IMVIP.2011.11.

[93] R. Dubé, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, C. Cadena, SegMatch: segment based loop-closure for 3D point clouds, CoRR (2016) abs/1609.07720. http://arxiv.org/abs/1609.07720.

[94] G. Klein, D. Murray, Improving the Agility of Keyframe-Based {SLAM}, in: Proc. 10th European Conference on Computer Vision, ECCV, 2008, pp. 802–815.

[95] B. Micusik, H. Wildenauer, Descriptor free visual indoor localization with line segments, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2015, pp. 3165–3173. http://dx.doi.org/10.1109/CVPR.2015.7298936.

[96] A. Vakhitov, J. Funke, F. Moreno-Noguer, Accurate and Linear Time Pose Estimation from Points and Lines, Springer International Publishing, Cham, 2016, pp. 583–599. http://dx.doi.org/10.1007/978-3-319-46478-7.36.

[97] B. Verhagen, R. Timofte, L.V. Gool, Scale-invariant line descriptors for wide baseline matching, in: IEEE Winter Conference on Applications of Computer Vision, WACV, IEEE, 2014, pp. 493–500.

[98] G. Yammine, E. Wige, F. Simmet, D. Niederkorn, A. Kaup, Novel similarity-invariant line descriptor and matching algorithm for global motion estimation, IEEE Trans. Circuits Syst. Video Technol. 24 (2014) 1323–1335. http://dx.doi.org/10.1109/TCSVT.2014.2302874.

[99] A. Concha, J. Civera, (2014) Using superpixels in monocular SLAM, in: 2014 IEEE International Conference on Robotics and Automation, ICRA, 2014, pp. 65–372. http://dx.doi.org/10.1109/ICRA.2014.6906883.

[100] J. Martinez-Carranza, A. Calway, Unifying planar and point mapping in monocular SLAM, in: Proceedings of the British Machine Vision Conference, BMVA Press, 2010, pp. 43.1–43.11. http://dx.doi.org/10.5244/C.24.43.

[101] D. Gálvez-López, M. Salas, J.D. Tardós, J.M.M. Montiel, Real-time monocular object SLAM, CoRR (2015) abs/1504.02398. http://arxiv.org/abs/1504.02398.

[102] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, F. Moreno-Noguer, Discriminative learning of deep convolutional feature point descriptors, in: Proceedings of the International Conference on Computer Vision, ICCV, 2015.

[103] Y. Verdie, K.M. Yi, P. Fua, V. Lepetit, TILDE: A temporally invariant learned detector, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2015, pp. 5279–5288.

[104] X. Han, T. Leung, Y. Jia, R. Sukthankar, A.C. Berg, MatchNet: Unifying feature and metric learning for patch-based matching, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2015, pp. 3279–3286. http://dx.doi.org/10.1109/CVPR.2015.7298948.

[105] S. Zagoruyko, N. Komodakis, Learning to compare image patches via convolutional neural networks, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2015, pp. 4353–4361. http://dx.doi.org/10.1109/CVPR.2015.7299064.

[106] K.M. Yi, Y. Verdie, P. Fua, V. Lepetit, Learning to assign orientations to feature points, in: Proceedings of the Computer Vision and Pattern Recognition, 2016.

[107] S. Pillai, J.J. Leonard, Monocular SLAM supported object recognition, CoRR abs/1506.01732 (2015). http://arxiv.org/abs/1506.01732.

[108] A. Kundu, Y. Li, F. Dellaert, F. Li, J. Rehg, Joint semantic segmentation and 3D reconstruction from monocular video, in: D. Fleet, T. Pajdla, B. Schiele, T. Tuytelaars (Eds.), Computer Vision–ECCV 2014, in: Lecture Notes in Computer Science, 8694, Springer International Publishing, 2014, pp. 703–718. http://dx.doi.org/10.1007/978-3-319-10599-4.45.

[109] N. Fioraio, L.D. Stefano, Joint detection, tracking and mapping by semantic bundle adjustment, in: Computer Vision and Pattern Recognition, CVPR, 2013 IEEE Conference on, 2013, pp. 1538–1545. http://dx.doi.org/10.1109/CVPR.2013.202.

[110] S. Savarese, Y.-W. Chao, M. Bagra, S.Y. Bao, Semantic structure from motion with points, regions, and objects, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 00, 2012, pp. 2703–2710.

[111] S. Yang, Y. Song, M. Kaess, S. Scherer, Pop-up SLAM: Semantic monocular plane SLAM for low-texture environments, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2016.

Daniel Asmar is an Associate Professor in Mechanical Engineering at the American University of Beirut. Daniel received his Bachelors degree in Mechanical Engineering from the University of Waterloo in 1993. He later earned his Masters degree in Mechanical Engineering from the American University of Beirut in 2002, and his Ph.D. in Systems Design Engineering from the University of Waterloo in 2006. Daniels research interests are in Robotics and Computer Vision. Specifically, he has interests in visual perception, autonomous robot navigation and mapping, environment representation and recognition, augmentation techniques in archeology, and segmentation methods in Computer Vision. He has several publications in these areas in refereed journals and conference proceedings. Daniel is an ASME member, a senior member in IEEE, and the chair of the joint IEEE Lebanese chapter in Robotics and Automation, Instrumentations and Measurements, and Control Systems.

Georges Younes is currently pursuing a joint Ph.D. in Systems Design Engineering between the University of Waterloo and the American university of Beirut. He received his Masters degree in Mechanical engineering from the American University of Beirut in 2016. His main research interests are in Computer Vision and Robotics, specifically in Visual SLAM techniques.

Elie Shammas acquired a Ph.D. from the Mechanical Engineering Department at Carnegie Mellon University in March 2006. From 2007 till 2010, Elie worked as a research and development engineer at Hexagon Metrology where he developed kinematic calibration algorithms for high accuracy portable Coordinate Measuring Machines. Since 2010, Elie has been an Assistant Professor in the Mechanical Engineering Department at the American University of Beirut. Elie's research expertise span multiple disciplines such as robotic joint and mobile robotic platform design, nonholonomic motion planning, and kinematic calibration of robotic manipulators.

John S. Zelek is an Associate Professor in Systems Design Engineering at the University of Waterloo. Currently he also serves as the Associate Graduate Chair as well. He has published over 150 refereed papers and has helped create at least 3 spin off companies resulting from research in his lab. His research interests include autonomous robotics, SLAM, scene understanding, machine learning, augmentative reality, as well as assistive devices. He is also a IEEE member and a registered professional engineer in Ontario Canada.