A constant-time SLAM back-end in the continuum between global mapping and submapping: application to visual stereo SLAM

The International Journal of Robotics Research
XX(X):1–20
©The Author(s) 2015
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Francisco-Angel Moreno[1], Jose-Luis Blanco[2] and Javier Gonzalez-Jimenez[1]

Abstract

This work addresses the development and application of a novel approach, called Sparser Relative Bundle Adjustment (SRBA), which exploits the inherent flexibility of the relative BA (RBA) framework to devise a continuum of strategies, ranging from RBA with linear graphs to classic BA in global coordinates, where submapping with local maps emerges as a natural intermediate solution. This method leads to graphs that can be optimized in bounded-time even at loop closures, regardless of the loop length. Furthermore, it is shown that the pattern in which relative coordinate variables are defined among keyframes has a significant impact on the graph optimization problem. By using the proposed scheme, optimization can be done more efficiently than in standard RBA, allowing the optimization of larger local maps for any given maximum computational cost. The main algorithms involved in the graph management, along with their complexity analyses, are presented to prove their bounded-time nature. One key advance of the present work is the demonstration that, under mild assumptions, the spanning trees for every single keyframe in the map can be incrementally built by a constant-time algorithm, even for arbitrary graph topologies. We validate our proposal within the scope of visual stereo SLAM by developing a complete system that includes a front-end that seamlessly integrates several state-of-the-art computer vision techniques such as ORB features and bag-of-words, along with a decision scheme for keyframe insertion and a SRBA-based back-end that operates as graph optimizer. Finally, a set of experiments in both indoor and outdoor conditions is presented to test the capabilities of this approach. Open-source implementations of the SRBA back-end and the stereo front-end have been released online.

Keywords

Bundle adjustment, computer vision, relative coordinates, stereo vision, SLAM

1 Introduction

Bundle Adjustment (BA) is a well-known problem in computer vision (Triggs et al. 2000) that consists in finding an optimal estimation to the positions of a set of visual landmarks, the camera poses from where images were captured and, typically, their calibration parameters too. This problem, also known as *full SLAM* (Thrun et al. 2005) or *structure-from-motion*, is typically addressed by minimizing a single cost function that simultaneously reflects the mismatch between the landmarks observed in the complete sequence of keyframes and their predictions according to the camera pose estimates. Traditionally avoided in the mobile robotics community due to its high computational cost, BA has recently gained an immense popularity, e.g. (Konolige and Agrawal 2008; Kaess et al. 2011), due to new advances in sparse algebra (e.g. sparse Cholesky decomposition (Davis 2006), inexact Newton type algorithms (Agarwal et al. 2010)), becoming a promising alternative to filtering methods extensively employed for SLAM during previous years (Strasdat et al. 2012).

BA-based SLAM defines a graphical model where nodes are the unknowns and edges represent constraints between them. There are two kinds of unknowns: the location of a set of discrete entities observed by the sensor, called *landmarks* (LMs), and a small fraction of all the poses (e.g. of a vehicle, a camera or a hand-held sensor) along its trajectory, called *keyframes* (KFs). Assuming Gaussian errors for all the observations and undertaking a least-squares criterion for the above-mentioned minimization, the process can be shown to become the maximum likelihood estimator for the problem (Hartley and Kahl 2007).

Under the general denomination of BA, different parameterizations have been proposed in the technical literature leading to the so-called *global* and *relative* BA techniques (Triggs et al. 2000). In short, global BA (GBA) selects a single KF to be the *origin* or reference frame for the coordinates of all subsequent keyframes and landmarks. As a consequence, a global, self-consistent map is obtained at the cost of updating all variables at every time-step. On the other hand, in relative BA (RBA), the problem unknowns are all *relative* positions between KFs and LMs. Typically, LM

[1]MAPIR-UMA Group, Dept. Ingeniería de Sistemas y Automática. Universidad de Málaga, 29071 Málaga, Spain
[2]Engineering Dpt., University of Almería, Spain

Corresponding author:
Francisco-Angel Moreno, Dept. Ingeniería de Sistemas y Automática. Universidad de Málaga. ETSI Informática, Lab. 2.3.6i, Campus de Teatinos – 29071 Málaga (Spain).
Email: famoreno@uma.es

coordinates are defined with respect to the KF from where they were observed for the first time, while the pose of each KF is defined relative to its predecessor (Sibley 2009). The latter feature leads to graphical models where KFs form a (mostly) linear graph, hence the motivation of naming this strategy *linear RBA* in the following. Very recently, this paradigm of relative coordinates has also been extended to work on continuous-time SLAM (Anderson et al. 2015), with basis functions used as continuous approximations of the estimated trajectory instead of relying on a sparse set of KFs.

Despite some advances, like iSAM (Kaess et al. 2008), any approach to SLAM relying on global coordinates, such as GBA, exhibits an inherent unbounded growth of the matrices involved in the graph optimization as the explored area becomes larger and longer loops are closed. Therefore, in principle, none of those SLAM methods could be practical for any robot aimed at life-long exploration. RBA approaches aim at solving that scalability issue (Sibley, Matthies and Sukhatme 2010), since they only optimize a local region of the graph around the latest KF (i.e. the current camera pose), hence achieving a bounded computational cost by definition. However, as it will be discussed later, the information matrices involved in RBA formulation become less sparse than their GBA counterparts, hence preventing it to fully exploit sparse algebra techniques and, consequently, to achieve an optimal performance.

The present work claims and shows that the sparsity level in such matrices can be controlled by changing the way in which relative coordinates are defined, which in turn determines how nodes are linked to each other within the graph. In particular, the creation of conveniently-connected *submaps* leads to sparser matrices than those in standard RBA approaches, without compromising their scalability advantages with respect to their global counterpart. This flexibility allows us to derive a continuum of strategies from linear RBA to submapping with local maps. This is the essence of the so-called *sparser* relative bundle adjustment (SRBA) method, preliminarily introduced in (Blanco et al. 2013), and now presented here in the context of a complete SLAM framework, tested with real datasets.

SRBA advocates the creation, within the graph, of relative submaps whose origin KFs are mostly linearly connected, thus defining *shortcuts* between nodes. This introduces zero-blocks in the matrices involved in the optimization (i.e. Jacobians and Hessians), hence becoming sparser. By following the strategy of optimizing only the graph around the current KF, also found in linear RBA, SRBA ensures bounded-time graph optimization, even in the event of closing loops of arbitrary length. However, we claim that different connection patterns allow for the optimization of larger map areas for *any* fixed maximum topological distance from the current keyframe. Finally, a *shortest-path spanning tree* (required by graph search operations during optimization) is incrementally maintained up to a certain topological distance using the novel *constant time* algorithm which was also introduced in (Blanco et al. 2013).

In summary, the present work extends our preliminary results on SRBA and presents the following contributions:

- The SRBA bounded-time nature is demonstrated by providing detailed pseudo-code listings and thorough complexity analyses for the most important algorithms that are performed on the graph.
- We provide an experimental validation, with real datasets in different conditions, of the SRBA back-end as graph optimizer, applied to stereo visual SLAM.
- A front-end is developed that seamlessly integrates several state-of-the-art computer vision techniques, along with a decision scheme for keyframe insertion. This includes all the necessary elements to build, along with our SRBA back-end, a complete stereo visual SLAM system.

Regarding the latter point, the proposed front-end is constructed upon the following building blocks. Keypoint detection and description is accomplished through the ORB method (Rublee et al. 2011), which provides features invariant to scale and rotation with associated binary descriptors. A binary *bag of words* (Galvez-Lopez and Tardos 2012) for ORB descriptors is employed to assist data association by restricting the search area when looking for loop closures, hence boosting the overall performance. Ego-motion estimations between consecutive frames are computed through our visual odometry method proposed in (Moreno et al. 2013), a fast and reliable outlier detector based on robust kernels, as an efficient alternative to RANSAC. Finally, we have devised a decision scheme based on image similarity and geometrical distance between keyframes to decide when to add new keyframes to the graph, aiming to keep it as small as possible.

Our proposal is supported by a set of experimental results with synthetic and real images recorded in both, indoor and outdoor conditions. A video of such experiments can be watched at `https://goo.gl/1Ap4p9` and the source code is available on-line*.

2 Related Work

Bundle adjustment has undergone a rebirth as an attractive solution to the SLAM problem, thanks to the incorporation of recent sparse algebra techniques. This has turned it into a strong alternative to probabilistic filtering, which had become the standard for real-time vision-based SLAM and global localization during the last decades, leading to a long list of works which can be found elsewhere, e.g. (Moreno et al. 2009; Blanco et al. 2010; Wolf et al. 2002; Civera et al. 2007). A thorough comparison between filtering and BA solutions is presented in the works (Strasdat et al. 2010) and (Strasdat et al. 2012), ultimately advocating for the application of BA as it outperforms filtering methods in terms of accuracy per unit of computation time.

Different techniques have been proposed to reduce the computational burden of general BA solutions, which quickly increases as the size of the map grows. For instance, the well-known PTAM method (Klein and Murray 2007) separates camera tracking and mapping in two parallel threads and performs KF-based global BA by optimizing the graph over a subset of the N most recent camera poses,

*The C++ source code for the front-end and back-end can be found in `https://github.com/famoreno/srba-stereo-slam` and in `https://github.com/MRPT/srba`, respectively.

in order to achieve real-time performance. However, its effectiveness is limited to small indoor scenarios and does not scale well with large environments.

Another notable approach to SLAM is incremental Smoothing And Mapping (iSAM) (Kaess et al. 2008), where a factored representation of the approximate Hessian matrix is exploited to provide easy access to the marginal covariances needed for data association. This matrix is incrementally updated and maintained sparse by reordering the involved variables.

On the other hand, there are several proposals that adopt a relative formulation, often in combination with other techniques such as *submapping* or data marginalization, in order to reduce the problem complexity and to achieve a bounded-cost performance that can lead to on-line, real-time operation, as claimed in several works, e.g. (Eade and Drummond 2008; Konolige and Agrawal 2008; Sibley 2009; Sibley et al. 2009; Sibley, Matthies and Sukhatme 2010; Lim et al. 2011).

A hybrid metrical-topological world representation was successfully proposed in (Blanco et al. 2008) within a Bayesian framework, as a natural mechanism to deal with the map scalability issue. In the context of BA, the work in (Lim et al. 2011) proposes a hybrid metrical-topological representation of the map which provides scalability to the bundle adjustment problem. This proposal benefits from the topological map properties to allow for instant loop closures while metric locally consistent maps are maintained by embedding neighbor keyframes and landmarks into a single Euclidean space and optimizing over the submap. The submaps are then treated as rigid bodies in a global adjustment process that yields a globally consistent map, although this last step is not performed in real-time.

In (Konolige and Agrawal 2008), a *skeleton* is built from the camera and landmark relative poses, in order to represent a reduced system that approximates the full problem, leading to feasible solutions when dealing with large loop closures. This skeleton is formed by marginalizing features over camera poses and subsequently further reducing the latter.

The works in (Sibley 2009; Sibley et al. 2009) constitute the basis of the RBA approach developed in this paper, being further extended in (Sibley, Matthies and Sukhatme 2010), where a sliding window filter (SWF) based on a delayed state marginalization is proposed. Its operation ranges from EKF to full SLAM according to the window size, while performing both landmark and camera pose marginalization to achieve constant time operation, demonstrating similar convergence properties to the full batch solution and outperforming those from standard visual odometry. Nevertheless, it has been specifically devised for EDL (Entry, Descent and Landing) applications for autonomous landing vehicles and there is not any validation in more general situations as typical indoor and outdoor urban environments. These scenarios, though, have been addressed in (Sibley 2009) and specially in (Sibley, Mei, Reid and Newman 2010), where a large set of outdoor experiments have been conducted to demonstrate the scalability potential of relative approaches. Among them, it can be highlighted the estimation of a 121-km outdoor path from Oxford to London that includes the use of different forms of transport such as foot, bikes, trains and some others.

Recently, in (Mur-Artal et al. 2015) it has been presented a visual SLAM system which has some similarities with the present work. It also addresses BA by relying on ORB features for tracking, mapping and performing place recognition based on bag-of-words. Since it operates on monocular images, it is provided with an automatic procedure for map initialization that includes the parallel computation of both an essential matrix and a fundamental matrix to infer the relative pose between two frames, regardless of the type of scene. Once computed, one of them is heuristically selected and the pose is derived from it. Our work does not need such procedure as we employ stereo images that provide immediate 3D information for the map initialization. Apart from this, the main difference with our approach comes from the fact that they optimize a reduced graph, denoted by *Essential Graph*, which contains all the problem keyframes but a reduced number of edges. Our work, on the contrary, relies on the optimization of the local area of the graph, which includes the current and nearby submaps (up to a certain topological distance) and all their associated edges, building a locally consistent map.

In comparison to the above-mentioned methods, in this paper we adopt an approach that, instead of being attached to a single strategy, defines a continuum of solutions that range from global BA to linear RBA, e.g. (Sibley et al. 2009). This flexibility is derived from a submapping scheme that allows us to exploit more thoroughly the inherent sparsity of the system matrices without performing data marginalization. The so-called *sparser* RBA will be explained in Section 5.

3 System Overview

This section describes the proposed system built upon ORB keypoints and SRBA to perform visual SLAM. It can be split into a front-end, which is in charge of extracting and matching features from the input stereo images, performing visual odometry between consecutive time-steps and, if needed, finding correspondences between the current observation and the map, and a back-end that creates, manages and performs inference on the graphical representation of all the unknown relative camera poses and landmark positions as the robot navigates.

At the core of our proposed front-end is the ORB feature detector (Rublee et al. 2011). Relying on the FAST and BRIEF methods, ORB is not only a fast and reliable feature detector but it also provides binary descriptors for the keypoints, which are efficiently matched by measuring Hamming distances. Data association is aided by the management of an image database based on a binary bag of words (Galvez-Lopez and Tardos 2012). When data association is needed, the set of descriptors associated to the keypoints extracted in an image is employed to query the database looking for the most similar image stored in it, thus leading to the process of matching features between the current observation and the stored map.

Regarding the back-end, the proposed SRBA method models the entities in the SLAM problem as a graph, whose nodes are KFs and LMs while the edges symbolize the relative constraints between them. In order to estimate both the world structure and the KF poses, a nonlinear least-squares optimization process is carried out, which minimizes

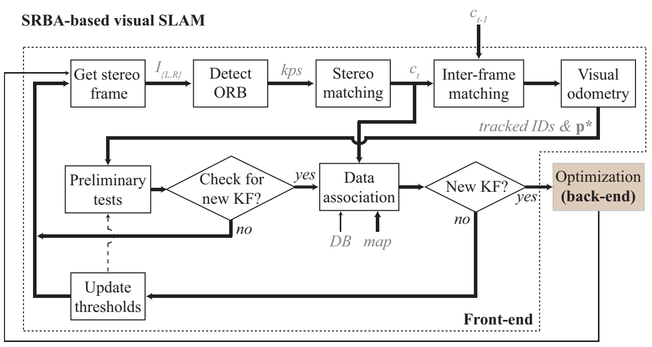

Figure 1. Scheme of the proposed SRBA-based visual SLAM system, with the focus on the front-end structure.

the mismatch between all the observed image features and their predicted positions in the image. Generally, such an error function is assumed to be quadratic in the image feature projection errors, though non-quadratic error functions, sometimes referred as *robust kernels*, can be chosen to deal with the presence of outliers in the observations. Huber kernels are typically used to this end, although recent quantitative benchmarking has demonstrated that other kernels achieve superior performance in photogrammetric and computer vision applications (Concha and Civera 2015). It must be noted that, by disregarding the estimation of landmark positions, we can follow a simplified version of the above optimization process to estimate the camera ego-motion between consecutive time-steps, i.e. to perform *visual odometry*.

Fig. 1 shows the general scheme of the proposed system, sketching all the stages from the acquisition of the stereo images to the joint estimation of landmark positions and robot poses up to the current time-step, carried out by the back-end. Note that the procedures included in our front-end have been emphasized in this figure, since they will be explained next. The back-end, represented here as a shaded block, is addressed later in section 5.

Notation

Formally, let us define the system state vector as:

$$\mathbf{s} = (\mathbf{p}, \mathbf{x}), \tag{1}$$

which encompasses the problem unknowns, i.e. the collection $\mathbf{p}$ of P keyframes and the set $\mathbf{x}$ of L 3D landmark positions:

$$\begin{aligned} \mathbf{p} &= \{\mathbf{p}_{j,b}\}_{j=1\ldots P,\ b\in\{1\ldots P\}\mid b\neq j} \\ \mathbf{x} &= \{\mathbf{x}_{j,b}\}_{j=1\ldots L,\ b\in\{1\ldots P\}}\ . \end{aligned} \tag{2}$$

For the KFs, this notation defines j as an identifier for the keyframe while b indicates the index of the reference frame which it is referred to. Note that b cannot be the same as j since it does not make sense to refer a keyframe to its own reference system. Regarding landmarks, j stands again for an identifier while b specifies the so-called *base* KF, which is the reference frame in which the j-th landmark coordinates are expressed. It is important to remark that the choice of *which* keyframe will be the reference keyframe b represents the main difference between global and relative approaches: i.e. in global SLAM there exists a privileged coordinate frame that serves as a reference for the entire map.

Let $\mathbf{z}$ be the set of N observations gathered at a certain time-step, each one assumed to be corrupted by zero-mean Gaussian noise with information (inverse covariance) matrix Λ_j:

$$\mathbf{z} = \left\{\mathbf{z}_j^o\right\}_{j=1,\ldots,N;\ o\in\{1,\ldots,P\}}, \tag{3}$$

where j represents the index of the observed landmark, and o stands for the index of the current KF, which will be called the *observer* KF.

In this paper, the observations are represented by 4D vectors comprising the image coordinates of the 3D landmark projections in both the left and right images:

$$\mathbf{z}_j^o = (u_L, v_L, u_R, v_R)_j^{\top}. \tag{4}$$

Finally, let $\mathbf{T}_{b,o} \in \mathbf{SE}(3)$ denote a 4×4 transformation matrix that represents the pose of the o-th KF with respect to the b-th KF, its reference system. Thus, the conversion of the j-th landmark coordinates, referred to its base KF b, to its coordinates with respect to a certain observer KF o is performed through:

$$\mathbf{x}_{j,b}^o = \left(\mathbf{T}_{b,o}\right)^{-1} \mathbf{x}_{j,b}. \tag{5}$$

In relative formulations, this matrix is built from a chain of transformations, becoming the main responsible for the loss of sparsity — we continue this discussion in Section 5.

For the sake of clarity, the subscript and superscript concerning the base and/or the observer KFs for the keyframes, the landmarks and/or the observations will be omitted whenever they are unnecessary.

4 Front-end

This section describes the methods implemented within the system front-end, from keypoint detection in a pair of stereo images to the extraction of a set of associated visual features between the current observation and the stored map.

4.1 ORB keypoints

Due to computational efficiency requirements, feature-based visual SLAM frameworks tend to rely on efficient methods to detect keypoints and extract descriptors that are fast to both compute and match. As a result, *binary* detectors and descriptors are becoming a standard. In this paper we employ the OpenCV (Bradski 2000) implementation of the ORB (Rublee et al. 2011) keypoint detector and descriptor.

Built upon the FAST detector (Rosten and Drummond 2006) and the BRIEF descriptor (Calonder et al. 2010), ORB enhances them by providing scale and rotation invariance to the detected keypoints. Stereo matching is subsequently performed through the Hamming distance (Hamming 1950) between ORB descriptors, i.e. counting the number of different bits between them. We also consider epipolar restrictions which, in this case, reduce to corresponding points lying on the same image row. The set of stereo matches at t-th time-step is denoted by c_t.

Finally, in order to provide tracking information to later stages in the system, a unique ID is assigned to every stereo match in c_t.

4.2 Visual odometry

Camera ego-motion between the previous time-step and the current one is then estimated by the visual odometry method we formerly presented in (Moreno et al. 2013).

First, *inter-frame* keypoint correspondences are found between the current stereo matches and the previous ones (c_t and c_{t-1} in Fig. 1, respectively). Following a similar approach than that for stereo matching, we look for potential correspondences between the current left image and the previous one, relying again on descriptor distances and epipolar restrictions (which, in this case, involves the computation of the fundamental matrix for the considered images). For the sake of robustness, this process is repeated for the right images so that only the matches consistent with those found in the left ones are kept.

Visual odometry is then computed following the general approach of iteratively minimizing a certain cost function which measures at each iteration the mismatch between the predicted and the observed image projections of corresponding keypoints at both time-steps:

$$F(\mathbf{p}) = \sum_i \frac{1}{2} \mathbf{\Delta z}_i^{\mathsf{T}} \mathbf{\Lambda}_i \mathbf{\Delta z}_i, \tag{6}$$

where $\mathbf{p}$ stands for the estimation of the camera pose change between time-steps $t-1$ and t and represents the only unknown within the system state in Eq. (1) for visual odometry. In this situation, therefore, the keypoints 3D coordinates are considered to be fixed, and are computed by back-projecting their image coordinates at time $t-1$.

On the other hand, $\mathbf{\Delta z}_i = \bar{\mathbf{z}}_i - \mathbf{z}_i$ stands for the error between the prediction $\bar{\mathbf{z}}_i$ of the i-th feature and its observation $\mathbf{z}_i$. The *prediction function* determines the coordinates of a keypoint in the current image according to its 3D position $\mathbf{x}$, and the estimation $\mathbf{p}$ at the current iteration:

$$\mathbf{h}(\mathbf{p}, \mathbf{x}) = \bar{\mathbf{z}} = \{\bar{\mathbf{z}}_i\}_{i=1,\ldots,N}. \tag{7}$$

Finally, $\mathbf{\Lambda}_i$ in Eq. (6) is an appropriate weighting matrix, proportional to the inverse covariance of the zero-mean normally distributed noise (with a standard deviation of σ_p) that affects keypoint locations. Assuming identical error distributions for all detected features in both u and v directions leads to the simplification $\mathbf{\Lambda}_i = \left(1/\sigma_p^2\right)\mathbf{I}, \forall i$.

Typically, the cost function is minimized by computing small increments $\mathbf{\Delta p}$ that are added to the current estimation until convergence, yielding a final solution $\mathbf{p}^*$. These increments can be found by solving the well-known Gauss-Newton equation:

$$\begin{aligned} \mathbf{H}\mathbf{\Delta p} &= -\mathbf{g}, \\ \left(\mathbf{J}^{\mathsf{T}}\mathbf{\Lambda}\mathbf{J}\right)\mathbf{\Delta p} &= -\mathbf{J}^{\mathsf{T}}\mathbf{\Lambda}\mathbf{\Delta z}, \end{aligned} \tag{8}$$

where $\mathbf{J} = \partial \mathbf{h_z}/\partial \mathbf{p}$ stands for the Jacobian matrix of the prediction function, and $\mathbf{\Delta z}$ is a block-column vector containing the errors $\mathbf{\Delta z}_i$ of the individual observations.

However, the quadratic cost function in Eq. (6) is not robust against the presence of outliers, so that the minimization process may converge to an invalid solution. Thus, we follow the ERODE method presented in (Moreno et al. 2013) which, instead, proposes the use of a robust cost model based on the pseudo-Huber function (Huber et al. 1981):

$$F_r(\mathbf{p}) = \sum_i \frac{1}{2}\left[2b^2\left(\sqrt{1+\left(\frac{s_i}{b^2}\right)}-1\right)\right], \tag{9}$$

with b being a parameter which tunes the shape of the function and $s_i = \mathbf{\Delta z}_i^{\mathsf{T}}\mathbf{\Lambda}_i\mathbf{\Delta z}_i$. Using this function also implies a change in the Gauss-Newton expression shown in Eq. (8), which becomes:

$$\left(\mathbf{J}^{\mathsf{T}}\mathbf{\Lambda}\mathbf{J}\right)\mathbf{\Delta p} = -\rho'\mathbf{J}^{\mathsf{T}}\mathbf{\Lambda}\mathbf{\Delta z}, \tag{10}$$

where $\rho' = \{\rho'_i\}$ is a block-column vector with the derivative of the pseudo-Huber function for each observation:

$$\rho'_i = \frac{\partial}{\partial s_i}\left[2b^2\left(\sqrt{1+\left(\frac{s_i}{b^2}\right)}-1\right)\right] = \frac{1}{\sqrt{1+\frac{s_i}{b^2}}}. \tag{11}$$

Adopting this procedure, the contribution in the cost function of the large errors introduced by the outliers is significantly mitigated. This approach implies that all input data are considered in the minimization process but, on the other hand, there is no need to try different hypotheses of the model (as with RANSAC). With this method, the estimation process naturally converges towards the true solution and, after a few iterations, the outliers appear clearly visible in the vector of residuals so that we can remove them and, subsequently, refine the estimated solution to achieve higher accuracy.

It is important to remark that, although visual odometry is computed at every time-step, we follow the usual practice in the computer vision and SLAM literature of not creating keyframes so often, but only when a set of conditions are fulfilled, as explained next.

4.3 Keyframe Creation Decision

Creating new keyframes is a computationally expensive process. It implies inserting new variables and constraints in the graph, then optimizing the whole system or part of it. This is specially noticeable when working with real data, since the number of features detected in images may become considerably large. However, in our previous work (Blanco et al. 2013) little attention was paid to this issue, since we focused on synthetic, sparse maps. In this work, on the contrary, a set of heuristics has been defined to decide when to create new keyframes, following the common idea of inserting a new one only when the explored area is becoming different enough from the last stored keyframe.

Algorithm 1 summarizes our proposed scheme for deciding when to create keyframes. The process starts from the visual odometry output, which consists of: (i) a list comprising the IDs of the tracked keypoints (assigned during stereo matching) and (ii) the estimation of the pose change between consecutive time-steps $\mathbf{p}^*$. The former is employed to keep track of the IDs assigned to the keypoints that have been correctly associated between consecutive time-steps. The drop of the number n_t of tracked keypoints below a certain threshold indicates that the observed scene is becoming significantly different. The latter, in turn, is employed to incrementally build a vector $\mathbf{c}_p$ that stands for the transformation between a certain camera pose and the one at the current time-step. Thus, as the camera

Algorithm 1 New keyframe decision scheme

```
Input:
    $\mathbf{IDs} = \{ID_1, \ldots, ID_{n_t}\}$    ▷ IDs of the $n_t$ tracked keypoints
    $\mathbf{p}^*$    ▷ Estimated pose change from visual odometry
    $\mathbf{c}_p$    ▷ Accumulated pose change, updated from the last call to this algorithm
Output:
    $d$    ▷ Decision about whether or not create a new keyframe
    $\mathbf{c}_p$    ▷ New accumulated pose change

1: $\mathbf{c}_p \leftarrow \mathbf{c}_p \oplus \mathbf{p}^*$    ▷ Update the current estimated pose change

2: if $|\mathbf{c}_p| > th_1$ OR $n_t > th_2$ then
3:     $\mathbf{IDs} \leftarrow \emptyset$    ▷ Reset tracked IDs
4:     $\mathbf{c}_p \leftarrow \mathbf{O}$    ▷ Reset to a zero pose change
5:     $d \leftarrow$ true    ▷ Create a new keyframe
6: else
7:     $d \leftarrow$ false    ▷ Do not create a new keyframe
8: end if
   return $\{d, \mathbf{c}_p\}$
```

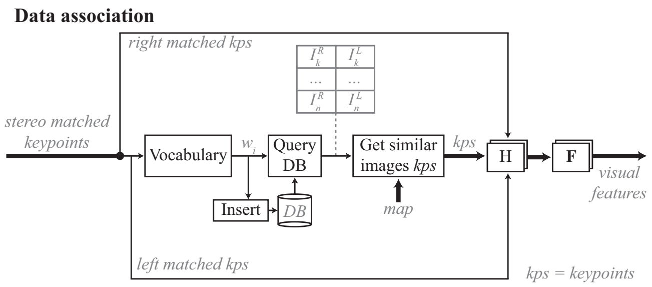

Figure 2. Scheme of our proposed data association procedure. Matched ORB keypoints (*kps*) from the left image are converted into a vocabulary word w_i employed to query (and which is subsequently inserted into) the image database (DB), retrieving a list of similar images. Then, current left and right keypoints are matched against those from the retrieved similar images by means of the Hamming distance between their descriptors (H blocks). Fundamental matrices $\mathbf{F}$ are finally employed to discard outliers.

moves, $\mathbf{c}_p$ incrementally grows in terms of translation and/or rotation. When such translation or rotation grow above some pre-defined thresholds, the camera is considered to have substantially moved from the previous keyframe and, therefore, it might be observing a significantly different area.

Any of these two indicators will trigger a data association process that will ultimately confirm whether or not the current observation has little in common with the system knowledge of the environment. In any case, both the monitored incremental pose $\mathbf{c}_p$ and the IDs of the tracked keypoints are reset, as the decision scheme will be started again from the beginning. Data association is discussed in Section 4.4.

In this paper, the number of associated visual features yielded by data association is employed as a measure of the similarity between the compared keyframes. Thus, if such number falls below a certain threshold (refer to Fig. 1), a new keyframe is created with the information extracted from the current pair of images. Otherwise, the present observation is considered to still share sufficient information with the most similar keyframe observed up to the current time-step, hence discarding the addition of a new keyframe. Finally, the thresholds employed in the preliminary tests are adjusted taking into account the number of associated visual features. Hence, the translation and rotation limits allowed before triggering a new data association process reduce as the measured similarity decreases.

4.4 Data Association

One of the key stages in the design of SLAM systems is the so-called *data association*. This term stands for the process of looking for correspondences between the current observation and the knowledge that the system has about the environment, i.e. the map. We refer interested readers to a more in-depth discussion of the problem presented elsewhere (Blanco et al. 2012). A special situation of paramount importance arises when the current observation matches to landmarks stored some time ago, hence implying a re-observation of an already explored area. This situation is known as *loop closure*.

In this work, data association follows the approach of the above-explained inter-frame matching process and extends it by introducing a previous stage that selects the most similar KF to the current one among those already stored in the graph. This stage aims at detecting loop closures and also relies on the ORB descriptors computed during keypoint detection to identify the corresponding observed landmarks, as depicted in Fig. 2.

We adopt a bag of words approach based on such binary descriptors to efficiently cope with this problem. The basic technique underlying a bag of words approach consists in building a database from the images recorded as the camera moves, then finding the most similar one within the database when a new image is captured. Specifically, we employ the method presented in (Galvez-Lopez and Tardos 2012), which was initially developed for BRIEF descriptors, but that has also proven to perform well with ORB descriptors because of their similarity.

In short, the bag of words technique uses a previously built visual vocabulary to transform an image to a numerical vector that is subsequently employed to look up among the images stored in the database. Since the bag of words presented in (Galvez-Lopez and Tardos 2012) is hierarchical, the vocabulary structure becomes a tree.

A schematic view of the different stages carried out in this paper to perform robust and reliable data association is shown in Fig. 2. There, it can be seen how, by using the pre-built word vocabulary, all the ORB descriptors extracted from an image (we only employ the left image) are summarized into a single word w_i that is subsequently stored within the database. As new images are captured, the so-obtained words are employed to query the database, retrieving this way a small list containing the most similar images within it. Therefore, the query result restricts the search for correspondences between individual landmarks to only those belonging to the most similar images instead of the whole map.

Once the most similar stereo images are retrieved, their keypoints and those from the current image pair are matched following the inter-frame matching process explained in Section 4.2, yielding a set of associated visual features which will represent the input of the SRBA-based back-end.

Finally, in order to consider an association as valid, all the matched projections of the landmark in the four images must be consistent with the epipolar geometry and the distance between their descriptors must fall below a threshold. This methodology reduces the presence of outliers in the data association output.

It is important to note that, as the camera explores unknown areas, the most similar image will be the last one inserted into the database, but, when visiting already explored zones, older images will be also retrieved, hence detecting loop closures.

5 Back-end

As already introduced in previous sections, one can establish two main variations of Bundle Adjustment (BA) techniques, also known as *smoothing* methods (Strasdat et al. 2010), according to the representation chosen for keyframes and landmarks: Global BA (GBA) and Relative BA (RBA). The principal differences between them is that GBA builds a global map by imposing a privileged KF to be the *origin* of coordinates, whilst RBA only estimates relative coordinates between KFs and LMs.

The main drawback of GBA that motivates the use of *relative* approaches to BA lies on the computational burden that global formulations suffer when dealing with large graphs, specially during loop closures. In theory, the number of unknowns that may need to be updated during a loop closure in GBA is *unbounded*. Relative methods, on the other hand, optimize only a small part of the graph, leading to smaller optimization problems which can be handled efficiently in *bounded* time. Though, the matrices associated with those smaller problems are significantly denser than their global counterpart due to the need of performing chains of pose compositions at each observation, since the coordinates of the stored landmarks (referred to their base-KF) have to be transformed to the current observer KF.

There exist two important concepts which are unique features of RBA and do not have a counterpart in GBA or global SLAM, namely:

- Maximum topological distance (D_{max}) for local map optimization. This parameter defines how far from the current KF should we optimize at each time-step, i.e. it determines the extension of the ”active part” of the map. Implicitly, this parameter also sets the minimum distance between the base KF of observed landmarks and its current observer KF for it to be considered a loop closure.
- Shortest-path spanning trees (STs) for every KF. These are data structures required to efficiently find the shortest paths between any pair of KFs within a distance D_{max}. These paths are needed to determine the variables involved in the chains of pose compositions employed to transform landmark coordinates and to determine what observations may be considered loop closures. A constant-time algorithm for incrementally updating STs over time was proposed in (Blanco et al. 2013) and will be summarized later.

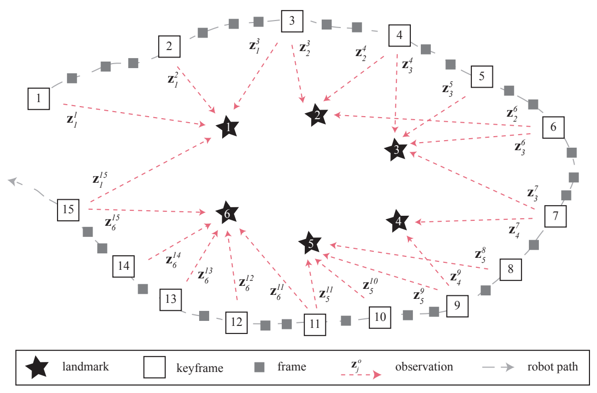

Figure 3. Example of a bundle adjustment problem.

As a consequence of optimizing only a part of the graph, RBA approaches create maps that are only locally consistent. Nevertheless, we claim that local maps of an environment, which include a correct representation of its global topological structure, are perfectly suitable for most common mobile robot operations, such as autonomous navigation via path planning and obstacle avoidance. Regarding the claims made elsewhere (Kaess et al. 2012) about the unsuitability of relative graphs for finding potential *shortcuts* during exploration, we believe that, provided that errors in the estimated relative poses between local maps are reasonably reduced, relative coordinates have enough accuracy to perform such operations. Notice that relative coordinates for KFs farther than D_{max} can be obtained at any instant by simply chaining the poses of edges along a spanning tree rooted at some KF in the area of interest. That is, any efficient global SLAM method (such as iSAM2 in (Kaess et al. 2012)) can be used to retrieve the global map from the relative graph built by RBA, although it would normally not be required for any robot operation.

Sparser Relative Bundle Adjustment (SRBA) (Blanco et al. 2013) was proposed as a blended solution in between GBA and the linearly connected implementation of RBA described above. By changing the policy about how to create new edges, SRBA introduced the idea of defining some KFs as *origin* KFs, dynamically selected as local reference systems and effectively creating *submaps*. Poses of all KFs within a submap are referred to its origin KF, whereas landmark coordinates are referred to the KF where they were observed for the first time, as usual in RBA. The underlying idea of this submap-based representation is to generate edges between origin KFs so that paths between KFs become shorter in average, thus reducing the loss of sparsity that linear RBA incurs in, while keeping its advantages over global mapping.

In order to illustrate the differences between the aforementioned approaches, we refer to the example scenario represented in Fig. 3. There, a robot follows an arbitrary path through an environment while gathering a total of 22 observations (shown as red-dashed arrows) of the 6 landmarks present in the scenario (displayed as numbered stars). In the figure, the observations are labeled following the notation introduced in Eq. (2). During navigation, 15 KFs have been defined among the whole set of captured frames.

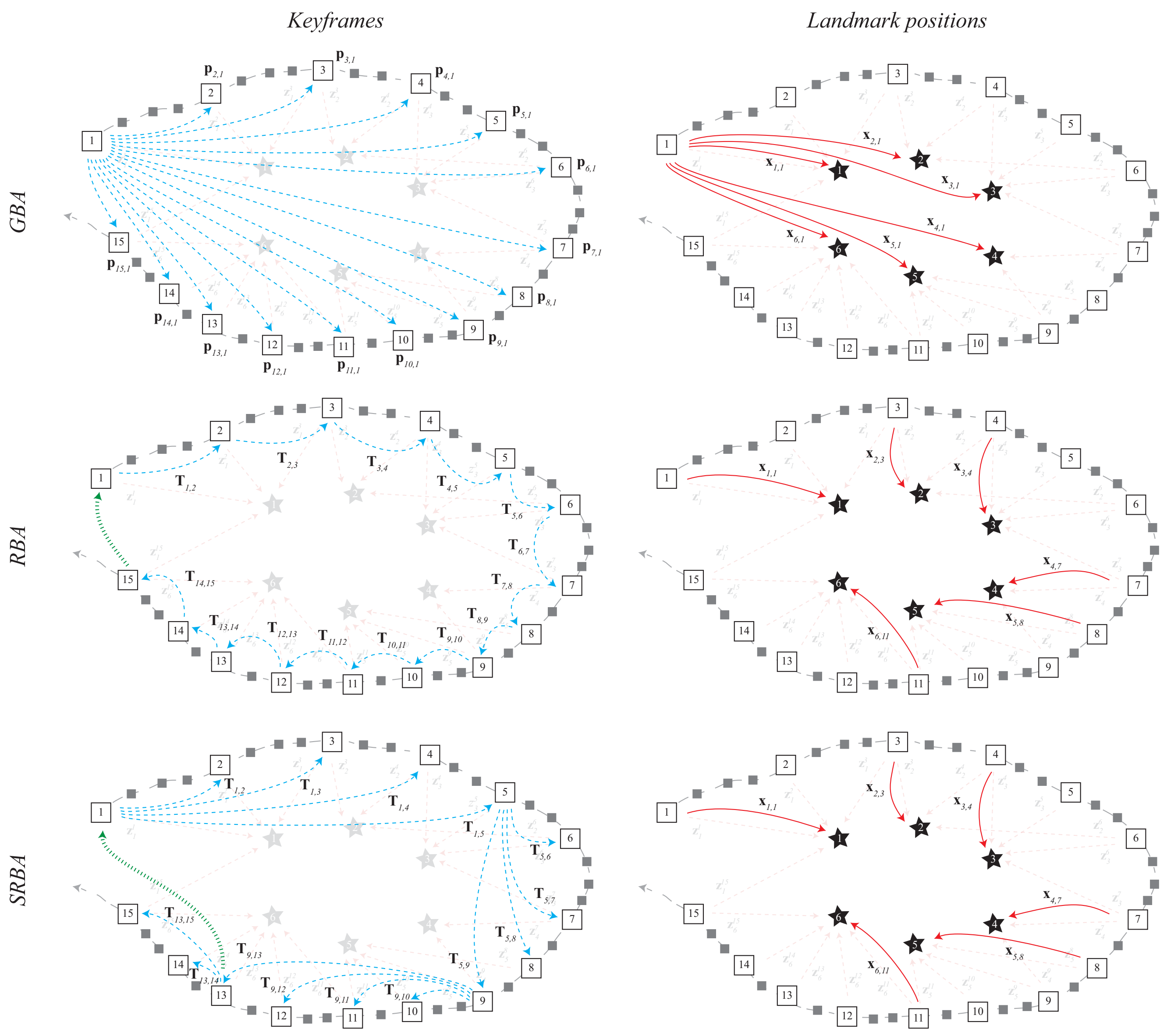

Figure 4. Graph representation under a GBA (top), linear RBA (middle) and SRBA (bottom) approaches. Blue-dashed arrows represent edges between KFs and their origins, while red-solid arrows indicate landmark positions with respect to their base KFs. For SRBA, the number of KFs within each submap has been set to 5.

In this situation, GBA defines all the KF poses and landmark positions referred to the global reference system centered at KF#1, as shown in Fig. 4 (top). Consequently, all the observations obtained from KFs different from KF#1 imply the transformation of the landmark global coordinates to the KF local reference system, hence inserting a non-zero block into the Jacobian matrix corresponding to the pose of the current observer KF. As a result of the global representation of KFs, this implies that $\mathbf{T}_{b,o}$ in Eq. (5) is composed by only one transformation with $b = 1$ for all the observations gathered from KFs different to the first one.

On the other hand, in linear RBA, the global poses of the keyframes are not the unknowns of the problem, but, instead, the relative transformations between them are. Usually, the defined edges link consecutive nodes creating linearly connected graphs, while landmarks are defined with their coordinates referred to the keyframe where they were observed for the first time, as shown in Fig. 4 (middle). In addition to that, another difference with GBA arises when the camera closes the loop, so that the problem structure must be modified due to the addition of a new unknown: the edge that joins the current KF and the *old*, re-visited one (please, refer to the green-dotted edge in Fig. 4 (middle) joining KF#15 and KF#1).

As a negative side effect, the sparsity level of both the Jacobian and the Hessian matrices results in fact reduced, degrading the efficacy of sparse algebra methodologies. This relative formulation leads to $\mathbf{T}_{b,o}$ in Eq. (5) becoming a chain of pose compositions following the path along the edges between the observer and the base keyframes, inserting dense blocks of size $L \times L$ in the matrices, with L being the number of edges between both KFs along the shortest path.

However, the potential of RBA lies on its flexibility, since it allows the optimization of only a subset of the problem unknowns, chosen according to the area affected by the current observation. This area is often called the *active region* and it is built through a breadth-first-search starting at current keyframe where the re-projection errors are computed in each keyframe, being added to the active region those with errors above a defined threshold. Moreover, those landmarks observed from the current keyframe are also considered to be part of the active region.

In addition to that, the active region is augmented by another set of keyframes called the *static region*, which is composed by any non-active keyframes that have measurements of the currently observed landmarks. The measurements at the static keyframes are included in the least-squares minimization but their relative positions are not optimized in the process, hence the term *static*.

Since only the unknowns involved in the active region are optimized in RBA, the Jacobian and Hessian matrices become significantly smaller and, more importantly, the complexity burden of the solution remains bounded even in the presence of large loop closures. Fixed and adaptive versions of the active region can be found in (Sibley 2009) and (Sibley et al. 2009), respectively.

Regarding SRBA, a possible graph created under this framework for such scenario is shown in Fig. 4 (bottom). Note how the graph has been divided in four submaps of size 5 whose origin KFs are those numbered 1, 5, 9 and 13. All the KFs within the submaps are referred to their origin KF and, similarly to the RBA case, their pose transformations with respect to it have been labeled by $\mathbf{T}_{j,b}$, with j standing for the KF index and b representing its reference frame (or origin KF). On the other hand, landmark positions $\mathbf{x}_{j,b}$ are represented following the same convention as for the standard RBA approach, with their coordinates referred to their base b KF. Both the poses of the KFs and the positions of the landmarks define the system state (i.e. the problem unknowns), to be determined from the observations, as in any bundle adjustment problem.

It can be seen in the figure that edges between origin KFs have been created (e.g. from KF#1 to KF#5) so that when the coordinates of a landmark have to be transformed from its base KF b (member of submap s_i) to the observer KF o (member of submap s_j), the corresponding chain of poses only involves the existing edges between the origin KFs of the traversed submaps (from s_i to s_j) in addition to those between KFs b and o and their respective submap origins. Generally, such path across the graph will be shorter than the chain of poses stated by standard linearly connected RBA, being this effect more beneficial for larger submaps. Thus, this approach may be intuitively understood as a way of creating *shortcuts* in the pose graph to shorten paths along KFs chains, therefore reducing L and leading to sparser matrices, as shown later.

It is important to note that, by setting up submap sizes from 1 to infinity, SRBA seamlessly integrates all the possible KFs configurations ranging from pure linear RBA to GBA, proving the flexibility of this blended approach.

5.1 Back-end implementation

As explained in Section 4.3, once the decision to insert a new keyframe has been taken, the set of visual features obtained through data association are fed to the SLAM back-end to maintain and optimize the graph that represents the relative map of SRBA.

The entry point for the back-end, summarized as pseudocode in Algorithm 2, takes the set of observations (and their data association) for the current KF, expands the graph as needed to accommodate the new KF, determines whether one or more KF-to-KF edges have to be created (the latter case corresponds to a loop closure) and optimizes the so-obtained nearby variables to ensure a consistent local map around the current KF. Subsequent subsections address, from a top-bottom perspective, each of the sub-algorithms required to perform all these tasks. Refer to Table 1 for a summary of the notation employed in all the algorithms. Worst case computational complexities are summarized along with the pseudo-code, in order to provide a validation of the bounded-time nature we claim for SRBA. Please, refer to (Blanco-Claraco 2013) for further low-level details about the C++ structures mentioned in these algorithms.

Table 1. Summary of notation employed in the algorithms

Symbol	Description
D_{max}	Maximum depth of maintained $\mathcal{ST}$s
N_R	The order of how many KFs are reachable within a range of D_{max}
N_o	Number of observations in a timestep
N_{obs}^{LC}	Minimum number of shared observations between two submaps to create a loop-closure edge
γ	Constant representing the expected "sparseness" of local maps
$\mathbf{B}$	Set of base KFs for all the observations in a KF
$\mathbf{C}$	Set of origin KFs for all submaps of $\mathbf{B}$
$\mathcal{ST}.\mathcal{D}[i][j]$	Spanning tree: Topological *distance* between KFs i and j. Symmetric table.
$\mathcal{ST}.\mathcal{N}[i][j]$	Spanning tree: The *next* edge to follow to reach j from i throughout the shortest path.
$\mathcal{ST}.\mathbf{Reach}[i]$	Spanning tree: The set of reachable KFs from i within D_{max}
$\mathcal{ST}.\mathbf{Seq}[i][j]$	Spanning tree: Sequence of all edges along the shortest path from i to j

From the computational cost analysis of our method, we can conclude that each time step has a typical cost that grows with the cube of N_R (the number of reachable KFs from any other KF within a maximum topological range of D_{max}) and linearly with N_o (the number of observed landmarks). Note that N_R is bounded, as long as it is the degree of the graph. A key issue here is modeling the number of edges (not nodes) within D_{max}. Assuming a constant ratio γ between edges and nodes, we have $O(\gamma N_R)$ edges to optimize in each time-step (line 8 in Algorithm 2).

To achieve minimal computational cost, N_R should be as reduced as possible, which involves having a sparsely connected graph. Nevertheless, the larger N_R is, a larger amount of keyframes and landmarks of the map are kept locally consistent. On the other hand, it is also desirable to decrease L (topological path length between observer KF and base KF) as much as possible, since it leads to smaller dense blocks in the approximate Hessian matrix. However this involves having a dense graph, which is in conflict with our first desiredatum.

As a trade-off, SRBA proposes a submapping approach. While *linear* RBA leads to linear graphs, ours generates a

```
Algorithm 2 srba_define_new_keyframe                    Worst case: $O((\gamma N_R)^3 + N_o(D_{max} + \log N_R))$
Input: $(\mathbf{z}_n, \alpha_n) = \left\{ \left(\mathbf{z}_n^1, \alpha_n^1\right), ..., \left(\mathbf{z}_n^{N_o}, \alpha_n^{N_o}\right) \right\}$    ▷ Set of $N_o$ new observations $\mathbf{z}_n^i$ and their data association $\alpha_n^i$
Output: The updated, locally consistent map

   // Update keyframes (KFs) data structures
1: $n$ ←number of KFs in the map                        ▷ Assign a free ID to the new KF – $O(1)$
2: $KF[n]$ ←empty KF data structure                     ▷ Insertion at the end of std::map – $O(1)$

   // Apply edge-creation policy to decide how to handle loop closures, etc.
3: edge_creation_policy($\alpha_n, n$)   (See Algorithm 3)        ▷ $O(|\mathbf{C}| N_R^2 \log N_R)$

   // Update symbolic Jacobian structures                  ▷ $O\left(N_o(D_{max} + \log N_R)\right)$
4: for each $\left(\mathbf{z}_n^i, \alpha_n^i\right) \in (\mathbf{z}_n, \alpha_n)$ do     ▷ For each of the $N_o$ new observations
5:     add_observation( $\underbrace{\mathbf{z}_n^i}_{\text{obs. data}}$ , $\underbrace{n}_{\text{observing KF}}$ , $\underbrace{\alpha_n^i}_{\text{landmark ID}}$ )   (See Algorithm 6)   ▷ $O(D_{max} + \log N_R)$
6: end for

   // Update SLAM estimation
7: edges_to_optimize ← all within a $D_{max}$ distance from $n$      ▷ $O(N_R)$
8: non_linear_optimizer(edges_to_optimize)                    ▷ $O((\gamma N_R)^3)$
```

hierarchical-like lay out. With this strategy we ensure that most KFs in the graph have a degree of one, while a large number of potential observations base KFs are still available within a short topological distance.

5.2 Edge creation policy: submapping strategy

The edge creation policy for the proposed SRBA formulation establishes a fixed number of KFs within the submaps so that when this limit is reached, a new submap is started with the new keyframe acting as its origin KF.

Algorithm 3 shows the process of deciding which edges must be created under such a fixed-size submapping approach. It should be mentioned that our open-source implementation allows users to redefine such *edge creation policy* (ECP) to ease further research.

In any case, a new edge e is always introduced between the last origin KF o_{local} and any new keyframe n which is not to become the origin of a new submap. Next, we search for additional edges by checking whether the visual features produced by the front-end contain a significant number of associations with landmarks whose base KFs are more distant to the current KF than D_{max}. In this case, the system determines that the camera is observing a far, already explored area, thereby detecting a loop closure and adding new edges e_{lc} to the graph (lines 25–27 of the pseudocode). We must note that there are no differences in the way that edges e and e_{lc} are inserted, no matter if they represent loop closure or simple constraints between origin KFs and submap members. With each sequential creation of new edges, the associated shortest-path spanning trees (STs) are updated to account for the changes introduced in the graph. Therefore, the order in which potential loop closure edges are considered (in the loop spanning lines 19–28) is important since, once an edge to a remote submap is added, other nearby submaps which were initially too far from the current KF (and could be interpreted also as loop closures), may fall within the D_{max} range, hence preventing the creation of additional edges. Thus, those submaps with the highest number of shared observations are considered first, since additional edges to them minimize the length of the pose chains from observer and base KFs, leading to more efficient nonlinear optimization steps.

In the pseudocode, this involves determining the set of base KFs associated to the current observations, as well as their multiplicity (i.e. the number of observations that belongs to each of them). Subsequently, these base KFs are grouped according to the submaps they are part of, with the aim of detecting the submaps that are affected by the current observation. Finally, we measure the distance between the origins of the involved submaps and the new node n, creating a new edge between them if such distance is over a certain value D_{max}, hence detecting the loop closure scenario.

The overall computational cost for this routine is $O(|\mathbf{C}| N_R^2 \log N_R)$, most of which emerges from the procedure for creating new edges, in Algorithm 4, with a cost of $O(N_R^2 \log N_R)$. Given that both $|\mathbf{C}|$ and N_R can be considered to be bounded under mild assumptions, it is clear that so are these algorithms.

5.3 Shortest-path spanning trees update

Regarding shortest-path spanning trees, the SRBA approach presented in (Blanco et al. 2013) creates and maintains STs for the graph up to a maximum topological distance of D_{max}, coinciding with the maximum map area that is also maintained locally consistent. There, the set of STs is implemented by means of two symbolic tables containing, for any two KFs i and j, both the topological distance in the graph between them, and the next node in the path from one to the other. Therefore, the shortest chain of poses between both KFs can be built by traversing these STs.

Although a ST for all the nodes in a graph could be easily created through classic breadth-first-search algorithms (Moore 1959), SRBA pursues the incremental build and update of both the graph and the STs as the environment is explored. Hence, the original paper proposed an incremental update algorithm for STs which we summarize next. When a new node, or *keyframe*, is added to the graph, a set of N edges connecting it to other existing keyframes must be defined (although usually it will be only one). Let n be the new node in the graph and i_k the target node for one

Algorithm 3 `edge_creation_policy` (Submapping strategy) Total: $O(|\mathbf{C}|N_R^2 \log N_R)$

```
Input:
    $\alpha_n = \{\alpha_n^1, ..., \alpha_n^{N_o}\}$                              ▷ IDs of all $N_o$ observed landmarks (Data association)
    $n$                                                                  ▷ ID of the new KF
Output:
    New edges added to the RBA graph

 1: $o_{local} \leftarrow submap\_id[n]$                                 ▷ Get ID of the current submap origin – $O(1)$
 2: $D_{LC} \leftarrow \underbrace{(D_{max}+1)}_{\text{Out of range}} - \underbrace{2}_{\text{Extra edges: } n \leftrightarrow o_{local}\text{, base KF to its origin}}$   ▷ Minimum distance between submap origins for loop closure – $O(1)$

    // Get ordered set of base KFs with observations in $\alpha_n$                ▷ Total: $O(N_o \log |\mathbf{B}|)$
 3: $\mathbf{B} \leftarrow \varnothing$,
    $\mathbf{B_M} \leftarrow 0$                                          ▷ Initialize set of KF observation bases and their multiplicity – $O(1)$
 4: for each $lm\_id \in \alpha_n$ do                                    ▷ $Total : O(N_o \log |\mathbf{B}|)$
 5:     $base\_kf\_id \leftarrow$get from $all\_lms[lm\_id]$             ▷ Random access to vector/deque container – $O(1)$
 6:     $\mathbf{B} \leftarrow \mathbf{B} \cup \{base\_kf\_id\}$         ▷ Insert in ordered set – $O(\log |\mathbf{B}|)$
 7:     $\mathbf{B_M}[base\_kf\_id] \leftarrow \mathbf{B_M}[base\_kf\_id] + 1$   ▷ Update KF multiplicity – $O(\log |\mathbf{B}|)$
 8: end for

    // Get ordered set of submaps with observations in $\alpha_n$             ▷ Total: $O(|\mathbf{B}| \log |\mathbf{C}|)$
 9: $\mathbf{C} \leftarrow \varnothing$,
    $\mathbf{C_M} \leftarrow 0$                                          ▷ Initialize set of KF submap origins and their multiplicity – $O(1)$
10: for each $base\_kf\_id \in \mathbf{B}$ do                            ▷ Total: $O(|\mathbf{B}| \log |\mathbf{C}|)$
11:     $s\_id \leftarrow submap\_id[base\_kf\_id]$                      ▷ $O(1)$
12:     $\mathbf{C} \leftarrow \mathbf{C} \cup \{s\_id\}$                ▷ Insert in ordered set – $O(\log |\mathbf{C}|)$
13:     $\mathbf{C_M}[s\_id] \leftarrow \mathbf{C_M}[s\_id] + \mathbf{B_M}[base\_kf\_id]$   ▷ Update multiplicity – $O(\log |\mathbf{C}|)$
14: end for
15: $\mathbf{C_S} \leftarrow sort(\mathbf{C}, \mathbf{C_M})$            ▷ Sort submaps by decreasing number of shared observations – $O(|\mathbf{C}| \log |\mathbf{C}|)$

    // Decide which KF-to-KF edges to create                             ▷ Total: $O(|\mathbf{C}|N_R^2 \log N_R)$
16: if $n \neq o_{local}$ then                                           ▷ If this KF is not the origin of the current submap
17:     create_kf2kf_edge($n \leftrightarrow o_{local}$)   (See Algorithm 4)   ▷ Always connect a KF to the origin KF of its submap – $O(N_R^2 \log N_R)$
18: end if
19: for each $o_{remote} \in \mathbf{C_S}$ such that $\mathbf{C_M}[o_{remote}] \geq N_{obs}^{LC}$ do   ▷ $O(|\mathbf{C}|)$ iterations. Total: $O(|\mathbf{C}|N_R^2 \log N_R)$
20:     if $o_{remote} \in \mathcal{ST}.\mathbf{Reach}[o_{local}]$ then  ▷ Find in a map container – $O(\log N_R)$
21:         $d \leftarrow \mathcal{ST}.\mathcal{D}[o_{local}][o_{remote}]$   ▷ Find in a map container – $O(\log N_R)$
22:     else
23:         $d \leftarrow \infty$                                        ▷ $O(1)$
24:     end if
25:     if $d \geq D_{LC}$ then
26:         create_kf2kf_edge($o_{local} \leftrightarrow o_{remote}$)   (See Algorithm 4)   ▷ Update KF-to-KF edge structures – $O(N_R^2 \log N_R)$
27:     end if
28: end for
```

Algorithm 4 `create_kf2kf_edge` Total: $O(N_R^2 \log N_R)$

```
Input:
    $f, t$                                                               ▷ IDs of the "from" and "to" KFs
    $\mathbf{p}_f^t$ (Optional; sensor model dependent)                  ▷ Initial value of the edge inverse pose

    // Allocate new kf-to-kf edge                                        ▷ Total: $O(1)$
 1: $new\_id \leftarrow$ number of entries in $kf2kf\_edges$             ▷ Query size of a deque container – $O(1)$
 2: Append new data structure with $\left(f, t, new\_id, \mathbf{p}_f^t\right)$ at the end of $kf2kf\_edges$   ▷ Insert at end of deque container – $O(1)$
 3: Append a *reference* to the end of $kf2kf\_edges$ to incidence lists for KFs $f$ and $t$   ▷ push_back to two containers – $O(1)$
 4: Append $new\_id$-th column to sparse Jacobian $\partial\mathbf{h}/\partial\mathbf{p}$   ▷ push_back to deque container of a symbolic CCS – $O(1)$

    // Update all nearby symbolic spanning trees
 5: update_sym_spanning_trees($f \leftrightarrow t$)   (See Algorithm 5)   ▷ $O(N_R^2 \log N_R)$
```

of the created edges, as shown in Fig. 5. The STs for all the nodes belonging to the STs of the involved n and i_k nodes must be checked for an update. This is performed by measuring the topological distance between any node r within the ST of n and any node s within the ST of i_k, which is determined by summing the distance between r and n (already stored in the ST of n), the edge $n \leftrightarrow i_k$ and the distance between the s and i_k (stored again in the ST of i_k). If n and i_k were not already in each other's ST and the resulting distance is D_{max} or less, both STs are updated to account for this new relation. Otherwise, if there was already a path between such nodes, their STs are only updated if the new path is shorter than the existing one. This process has to be repeated for all the created new edges, and it is summarized in Algorithm 5. A careful analysis of its computational cost, including access and creation of data structures, reveals a worst case complexity of $O(N_R^2 \log N_R)$, with N_R being the maximum number of reachable KFs for a given D_{max} and

```
Algorithm 5 update_sym_spanning_trees                          Worst case: $O(N_R^2 \log N_R)$
Input:
    $(i_k \leftrightarrow n)$                                      ▷ A new edge
    $D_{max}$                                                      ▷ The maximum desired depth of span. trees

 1: $ST^{i_k}_{D_{max}-1} \leftarrow \{\forall v / d(v, i_k) \leq D_{max} - 1\}$          ▷ $O(N_R)$
 2: $ST^{n}_{D_{max}} \leftarrow \{\forall v / d(v, n) \leq D_{max}\}$                    ▷ $O(N_R)$

 3: for each $r \in ST^{n}_{D_{max}}$ do                                                   ▷ $O(N_R)$ iterations
 4:     for each $s \in ST^{i_k}_{D_{max}-1}$ do                                           ▷ $O(N_R)$ iterations
 5:         // New tentative distance between $r$ and $s$
 6:         $d \leftarrow \mathcal{ST}.\mathcal{D}[n][r] + \mathcal{ST}.\mathcal{D}[i_k][s] + 1$   ▷ $O(\log N_R)$
 7:         if $(s \in \mathcal{ST}.\mathbf{Reach}[r]$ and $d < \mathcal{ST}.\mathcal{D}[r][s])$ or $(s \notin \mathcal{ST}.\mathbf{Reach}[r]$ and $d \leq D_{max})$ then   ▷ $O(\log N_R)$
 8:             // Shorter or new path found. Update trees:
 9:             $\mathcal{ST}.\mathcal{D}[r][s] \leftarrow d$
10:             $\mathcal{ST}.\mathcal{N}[r][s] \leftarrow \begin{cases} i_k & r = n \\ \mathcal{ST}.\mathcal{N}[r][n] & r \neq n \end{cases}$
11:             $\mathcal{ST}.\mathcal{D}[s][r] \leftarrow d$                                   ▷ $O(\log N_R)$
12:             $\mathcal{ST}.\mathcal{N}[s][r] \leftarrow \begin{cases} n & s = i_k \\ \mathcal{ST}.\mathcal{N}[s][i_k] & s \neq i_k \end{cases}$
13:         end if
14:     end for
15: end for
```

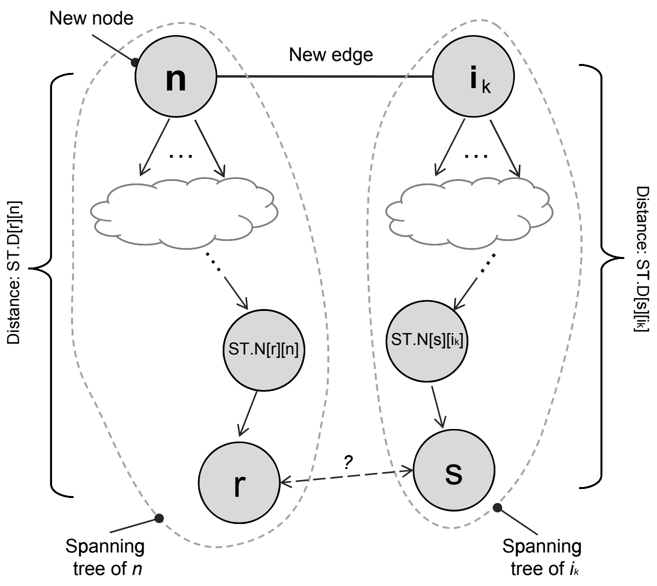

Figure 5. Diagram illustrating all the elements considered in the proposed algorithm for incremental update of STs. The creation of a new edge between n and i_k may become a shortcut that changes the shortest paths between nodes at each side of the new edge.

which is reasonable to consider bounded for any large and complex map, as long as redundant KFs are not continuously added to the map for the same physical area.

5.4 Observation insertion

Finally, lines from 4 to 6 in Algorithm 2 refer to the insertion of the current set of observations into the SRBA-based SLAM system, a process summarized in Algorithm 6.

For each individual observation, the general procedure involves creating and appending new structures to both the system observations and landmarks containers (the latter just in case we are observing a new landmark). This is performed in $O(1)$.

Subsequently, we proceed to update the system sparse Jacobian structure, which is stored in two separate parts: the Jacobian with respect to the edges ($\partial \mathbf{h}/\partial \mathbf{p}$) and with respect to the landmarks ($\partial \mathbf{h}/\partial \mathbf{x}$). For the former, we need to update all the matrix columns associated to the edges in the chain of poses between the current *observer* KF and the observation base KF, by appending a new block to each one of them (refer to Fig. 6 for an example of a Jacobian matrix). In this situation, the strategy followed by SRBA reduces the number of entries that must be updated since the traversed paths involves fewer edges, hence keeping the Jacobian sparser than standard RBA approaches. The complexity of updating this part of the Jacobian is $O(P + \log M)$, with P being the number of edges between the above-mentioned KFs and M the number of nodes stored in the system ST. In the worst case we have that $M = N_R$ and $P = D_{max}$, i.e. the number of reachable nodes in each ST and the maximum topological distance allowed for the ST, respectively.

Updating the Jacobian with respect to the landmarks is much simpler and can be performed in $O(1)$.

5.5 Least-squares optimization

The graph optimization is performed by an iterative Levenberg-Marquardt algorithm, which is a slight variation of the Gauss-Newton procedure that minimizes the re-projection error of the observed landmarks explained in section 4.2 for the visual odometry method. In this case, though, minimizing the cost function in Eq. (6) achieves the joint estimation of both the poses of the cameras $\mathbf{p}$ that captured the images (i.e. the keyframes) and the landmark positions $\mathbf{x}$. This leads to the following linear equation, equivalent to Eq. (8) but taking into account the whole system:

$$(\mathbf{H} + \lambda \mathbf{I}) \, \boldsymbol{\Delta}\mathbf{s} = -\mathbf{g}, \qquad (12)$$

where $\mathbf{H}$ and $\mathbf{g}$ stand for the approximate Hessian matrix and the gradient of the prediction function, respectively, $\boldsymbol{\Delta}\mathbf{s}$ standing for the incremental change in the system unknowns

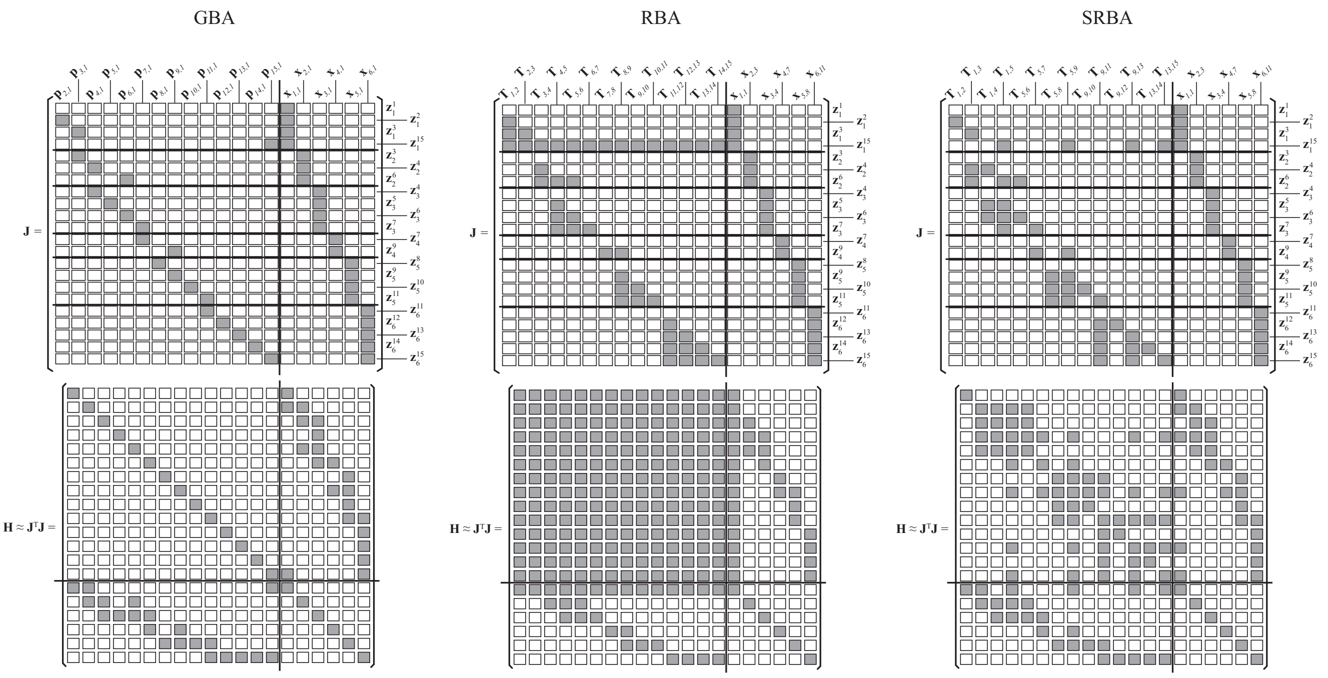

Figure 6. Sparsity pattern of Jacobian and Hessian matrices under the GBA, linear RBA and SRBA approaches for the scenario shown in Fig. 3. Gray squares represent dense blocks within the matrix.

Algorithm 6 `add_observation` Total: $O(D_{max} + \log N_R)$

```
Input:
    $\mathbf{z}$                                        ▷ Observation data
    $kf\_id$                                            ▷ ID of the observing KF
    $lm\_id$                                            ▷ ID of the observed landmark

    // Observation data housekeeping
 1: $obs\_id \leftarrow$ overall number of observations in $all\_obs$      ▷ Query size of a vector/deque container – $O(1)$
 2: Append $(\mathbf{z}, kf\_id, lm\_id)$ at the end of $all\_obs$         ▷ push_back on vector/deque container – $O(1)$

    // Handle new landmarks
 3: if $lm\_id \notin all\_lms$ then                                        ▷ Test for existence flag in a vector/deque container – $O(1)$
 4:     $all\_lms[lm\_id] \leftarrow$ new $kf2lm$ edge data structure        ▷ Create new $kf2lm$ edge – Amortized $O(1)$
 5:     Append column to sparse Jacobian $\partial\mathbf{h}/\partial\mathbf{x}$    ▷ push_back to deque container of a symbolic CCS – $O(1)$
 6:     Update bimap of column indices in $\partial\mathbf{h}/\partial\mathbf{x} \leftrightarrow$ landmark indices    ▷ With map_as_vector can be done in $O(1)$
 7: end if

 8: $base\_id \leftarrow$get from $all\_lms[lm\_id]$                        ▷ Random access to vector/deque container – $O(1)$
 9: Append a *reference* to $all\_lms[lm\_id]$ to the incidence list for $kf\_id$     ▷ $O(1)$

    // Update symbolic linear system ($\partial\mathbf{h}/\partial\mathbf{p}$ part)     ▷ Total: $O(D_{max} + \log N_R)$
10: if $base\_id \neq kf\_id$ then                                          ▷ Observations from the base KF introduce no blocks to this Jacobian
11:     $\mathbf{obs\_edges} \leftarrow \mathcal{ST}.\mathbf{Seq}\ \underbrace{[base\_id]}_{\text{Access in } O(1)}\ \underbrace{[kf\_id]}_{\text{Access in } O(\log N_R)}$    ▷ Retrieve list of $\leq D_{max}$ observed kf2kf edges from spanning trees – $O(\log N_R)$
12:     for each $kf2kf\_edge\_id \in \mathbf{obs\_edges}$ do                 ▷ $O(D_{max})$ iterations
13:         $\mathbf{col} \leftarrow [\partial\mathbf{h}/\partial\mathbf{p}]\ .\mathbf{cols}\ \underbrace{[kf2kf\_edge\_id]}_{\text{Access to deque in } O(1)}$    ▷ Retrieve symbolic sparse column – $O(1)$
14:         $\mathbf{col}\ \underbrace{[obs\_id]}_{\text{Insert into map}} \leftarrow$*reference* to observation data and other symbolic data    ▷ Typically, an insertion at the end of map – Typ. $O(1)$
15:     end for
16: end if

    // Update symbolic linear system ($\partial\mathbf{h}/\partial\mathbf{x}$ part)     ▷ Total: $O(1)$
17: $kf2lm\_edge\_id \leftarrow$ bimap from $lm\_id$                          ▷ With map_as_vector can be done in $O(1)$
18: $\mathbf{col} \leftarrow [\partial\mathbf{h}/\partial\mathbf{x}]\ .\mathbf{cols}\ \underbrace{[kf2lm\_edge\_id]}_{\text{Access to deque in } O(1)}$    ▷ Retrieve symbolic sparse column – $O(1)$
19: $\mathbf{col}\ \underbrace{[obs\_id]}_{\text{Insert into map}} \leftarrow$*reference* to observation data and other symbolic data    ▷ Typically, an insertion at the end of map – Typ. $O(1)$
```

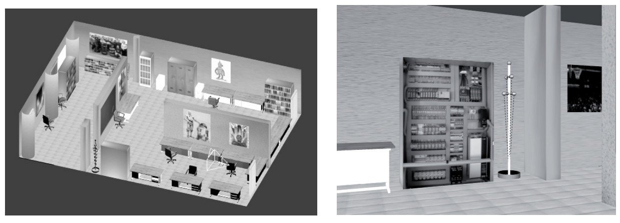

Figure 7. (a) Plan and (b) example image of the synthetic dataset.

$s = (\mathbf{p}, \mathbf{x})$ and λ a scalar parameter of the Levenberg-Marquardt algorithm. Notice that part of the state vector, in particular, all the relative poses, has the non-Euclidean topology of $\mathbf{SE}(3)$. Therefore, we have followed the common practice of formulating Eq. (12) in the error state formulation (e.g. see Grisetti et al. (2010)), such that increments on $\mathfrak{se}(3)$ are obtained from the linear solver, then added as corrections through the corresponding Lie group exponential map. For further mathematical details, the interested reader could refer to the technical report in Blanco (2010).

The sparsity of the Jacobian matrices, and hence the Hessian, plays here an important role in terms of computational burden, favoring SRBA by improving its performance in comparison to the standard, linear RBA method. Fig. 6 represents the sparsity patterns of such matrices for the example proposed in Fig. 3 under the GBA, RBA and SRBA formulations. In comparison to those for the GBA and standard RBA approaches, it can be noted that SRBA falls in between both formulations, yielding sparser matrices than RBA but denser than GBA.

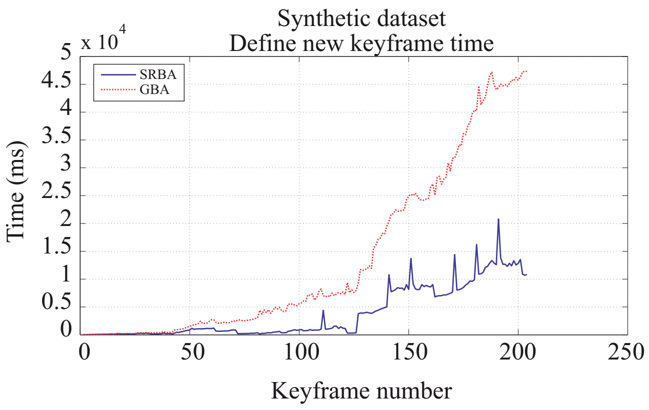

Figure 8. Inserting new keyframe time (including graph optimization) comparison for SRBA (blue-solid line) and GBA (red-dashed line) for the synthetic dataset.

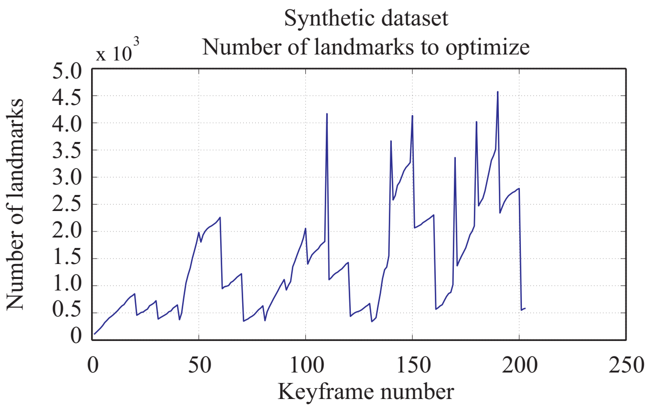

Figure 9. Number of landmarks to optimize with SRBA for the synthetic dataset.

6 Experimental Results

Experiments in both indoor and outdoor environments are presented in this section in order to validate the proposed system.

6.1 *Synthetic Images Dataset*

In our first experiment, we have moved a virtual stereo camera through a synthetic office-like environment created with the 3D design software Blender while capturing stereo images to form a dataset of synthetic images. A representation of such environment and an example of the captured images are shown in Fig. 7. The use of a virtual environment to test our approach is motivated here by the fact that the camera movement is known beforehand, hence making a *ground truth* available for the camera path.

As presented before, the main advantage of our relative representation is that the time spent in creating and inserting new keyframes (including the creation of the Hessian matrix and the subsequent optimization process) remains bounded, unlike the global approach. This is a consequence of the smaller matrices employed in our method due to the existence of submaps and a limit depth for graph optimization. To test this, we have employed the same dataset as input for both our SRBA-based approach and a pure global one (GBA).

In this sense, a performance comparison can be seen in Fig. 8, where the keyframe creation time (including the graph optimization step) is plotted for every keyframe defined during the experiment under both approaches. Please, note that we have not performed any optimization in our code to pursue efficient performance, hence the mentioned plot should be understood as a mere relative comparison between both techniques, disregarding the absolute time values. Fig. 9 also presents the number of landmark positions that are optimized at each keyframe insertion for the SRBA method, showing that it remains bounded over time, unlike global approaches.

Finally, we present in Fig. 10 a comparison between the estimated camera trajectory and the ground truth. The path of the camera has been obtained through a final full optimization of the graph built through our method in order to determine both keyframe poses and landmark positions within an unique global reference system whose origin is set at the first camera position. We also present in Fig. 11 the errors at the estimated camera positions associated to all the keyframes.

It has to be noted that, although the creation of an unique global map can not be considered part of our SRBA formulation, the presented path comparison and accuracy results represent indicators about the suitability of our method to store and manage enough information to create a global map in case that the application demands it. At the same time, it allows a more efficient way of estimating a camera trajectory while creating a map of the environment

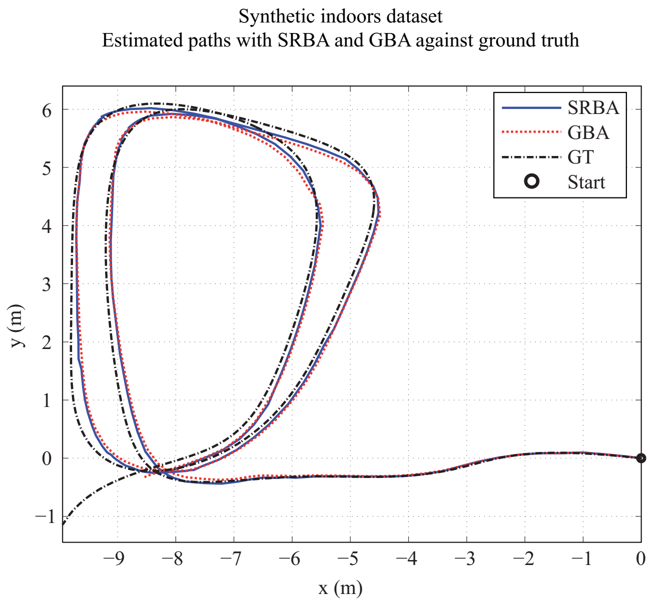

Figure 10. Estimated paths for the synthetic dataset with SRBA (blue-solid line) and GBA (red-dashed line). Ground truth is also represented (black dashed-dotted line).

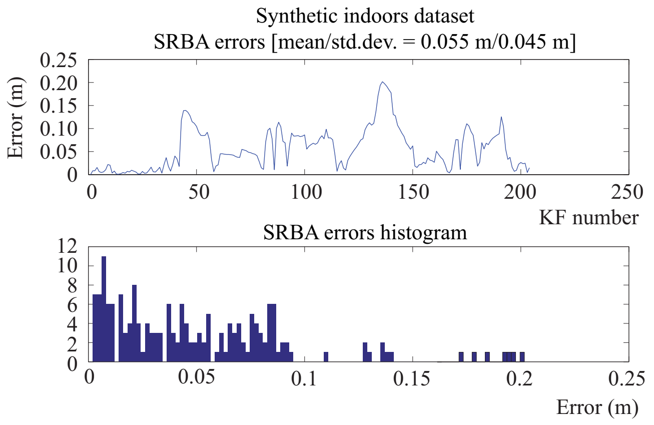

Figure 11. Absolute error and histogram of errors for our SRBA-based method with respect to the ground truth for the synthetic dataset.

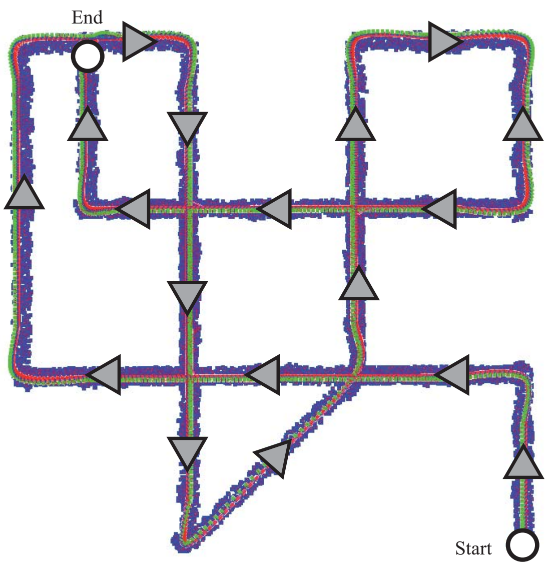

Figure 12. Path followed by the robot in the synthetic dataset used to compare SRBA with linear RBA. Observed visual landmarks are represented as blue dots.

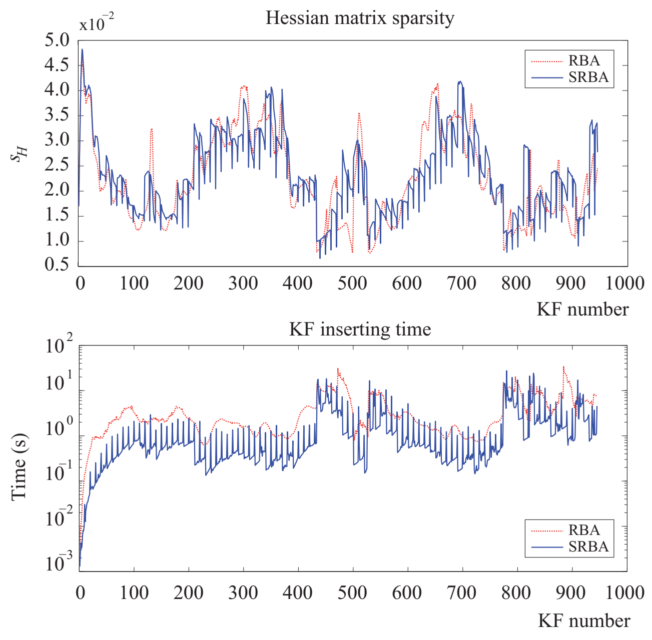

Figure 13. Performance comparison for classic (linear) RBA (dashed-red line) and SRBA (solid-blue line). (Top) Hessian matrix sparsity as the ratio of non-zero matrix blocks and (bottom) time spent in optimizing the graph for each KF insertion (y-axis is logarithmic). In both graphs, smaller is better.

which is locally consistent and useful enough for many applications.

Finally, aiming at comparing the performance of standard RBA and our approach, we present another experiment with a synthetic dataset generated with the freely available Recursive World Toolkit (RWT)[†]. In this case, we simulate the observations of a stereo camera moving along a corridor-like environment, which is populated with randomly-distributed 3D landmarks, and includes several loops, as shown in Fig. 12. We subsequently compare the sparsity of the Hessian matrix s_H at every keyframe insertion (and graph optimization), defined, in this experiment, as the ratio between the non-zero entries of the matrix and the total number of elements. This comparison, together with the time spent in the graph optimization for both methods, can be seen in Fig. 13. Note the higher matrix sparsity in the SRBA approach, leading to smaller computational times including at loop closures.

Moreover, we present in Fig. 14 the number of keyframes and landmarks that are updated in each keyframe insertion for both approaches, as a measure of the *size* of the local map that is updated, and hence maintained consistent, after each graph optimization. Notice that, overall, our proposal keeps consistency for larger local maps than linear RBA.

6.2 Real Indoors Dataset

In this second experiment we have tested our method against a real image dataset gathered with Sancho (Gonzalez et al. 2009), one of our mobile robots, while navigating through

[†]https://github.com/jlblancoc/recursive-world-toolkit

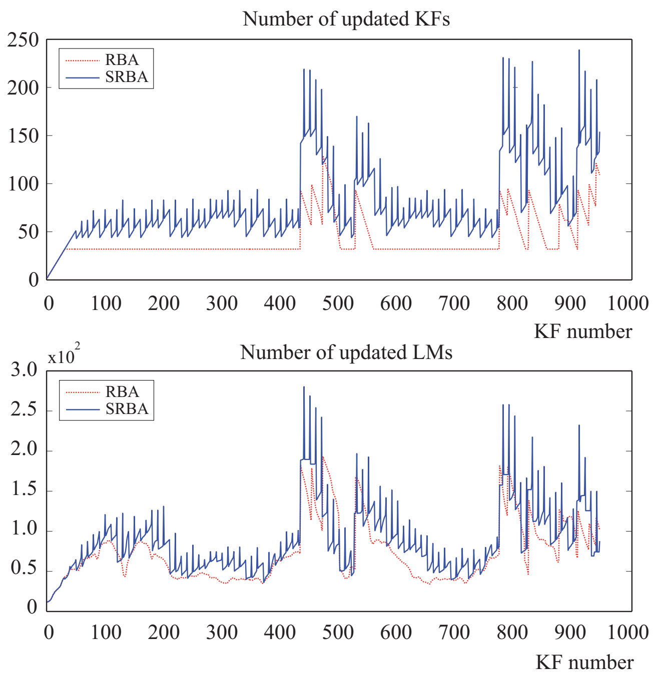

Figure 14. Number of (top) keyframes and (bottom) landmarks that are updated at each keyframe insertion (involving graph optimization) for standard RBA (dashed-red line) and SRBA (solid-blue line). In both graphs, larger is better.

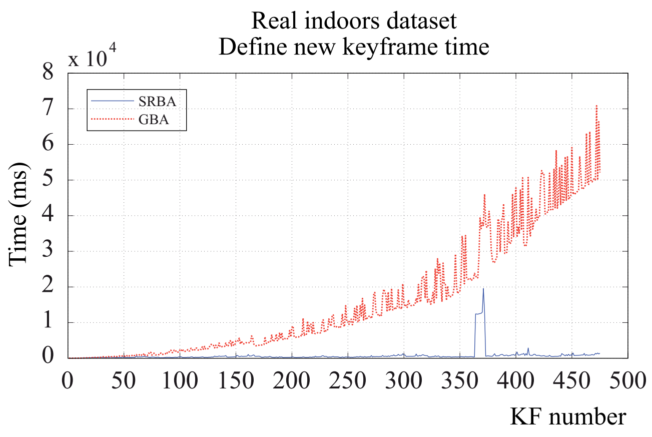

Figure 15. Inserting new keyframe time (including graph optimization) comparison for SRBA (blue-solid line) and GBA (red-dashed line) for the real indoors dataset.

Figure 16. Example images for the real indoors dataset.

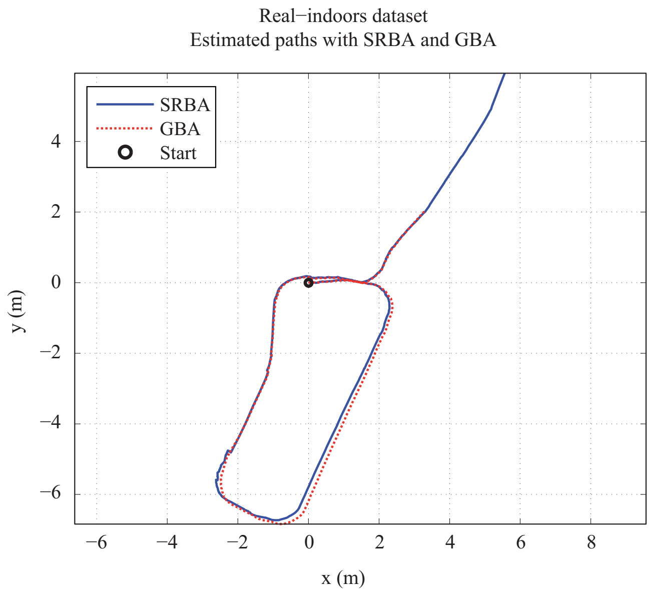

Figure 17. Estimated paths for the real indoors dataset with SRBA (blue-solid line) and GBA (red-dashed line).

our laboratory following a trajectory that includes a loop. Unfortunately no ground truth could be provided for this experiment since no other sensors where available on the robot at that time. Therefore both the obtained map and the robot path can only be validated by means of visual inspection.

Fig. 15 presents the timing comparison when creating new keyframes under both our SRBA-based approach and a global one. It can be noted a spike in SRBA insertion time around KF #360 due the detection of loop closure. At that time, the number of KFs included into the graph optimization increases significantly, since the *old* submaps involved in the closure are also included together with the most recent ones. Finally, regarding the time absolute values, similar considerations to those mentioned in the previous experiment must be taken into account here. An example of the images employed in this experiment can be found in Fig. 16 while Fig. 17 shows the estimated trajectory of the camera for both methods.

6.3 Real Outdoors Dataset

Finally, we have employed two outdoor datasets to test our method under more challenging conditions. In concrete we use one of our published stereo datasets (Blanco-Claraco et al. 2014) (referred as MAPIR dataset from now on), and one of the *road* datasets from the KITTI Vision Benchmark Suite (Geiger et al. 2012; Fritsch et al. 2013).

First, we have chosen the fragment number 7 of the MAPIR outdoors dataset, which contains a $\sim$0.7 km long loop within an urban scenario gathered by an on-board stereo camera placed on a standard car. The positioning information provided by a standard differential GPS device is considered the ground truth for this experiment.

The timing comparison when creating new keyframes for this experiment is shown in Fig. 18. Note that due to the size of the resulting map, with a total amount of more than 110k landmarks, the time spent when inserting keyframes by our GBA implementation rapidly becomes excessive. Therefore, beyond the insertion of keyframe #250, GBA time values are no longer shown in the plot. The small peak that appears around the keyframe #630 for the SRBA-method is due to the loop closure detection.

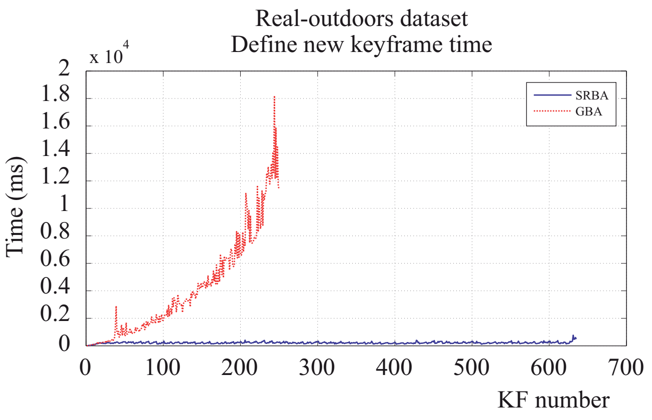

Figure 18. Inserting new keyframe time (including graph optimization) comparison for SRBA (blue-solid line) and GBA (red-dashed line) for the real outdoors dataset.

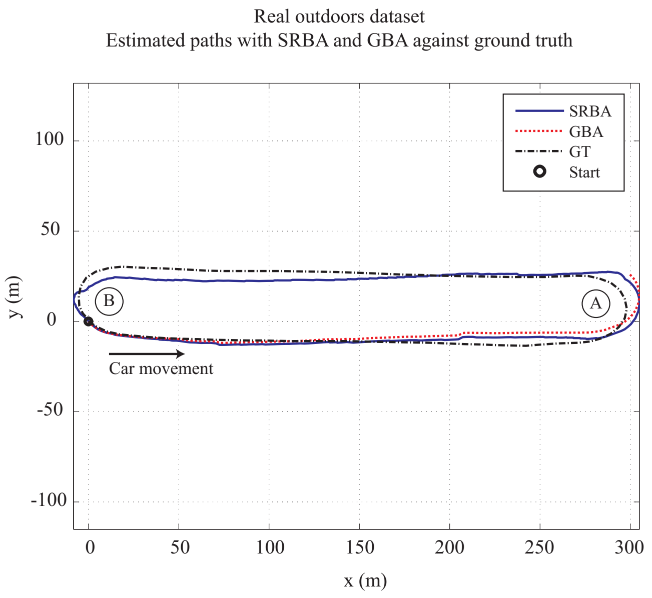

Figure 19. Estimated paths for the real outdoors dataset with SRBA (blue-solid line) and GBA (red-dashed line). Ground truth is also represented (black dashed-dotted line).

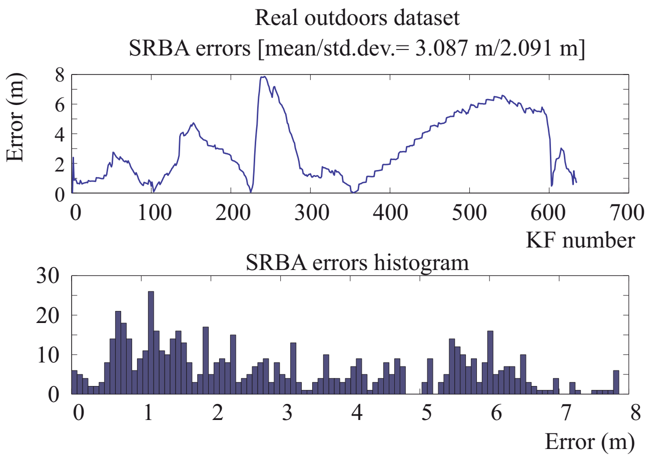

Figure 20. Absolute error and histogram of errors for the SRBA method with respect to the ground truth for the real outdoors dataset.

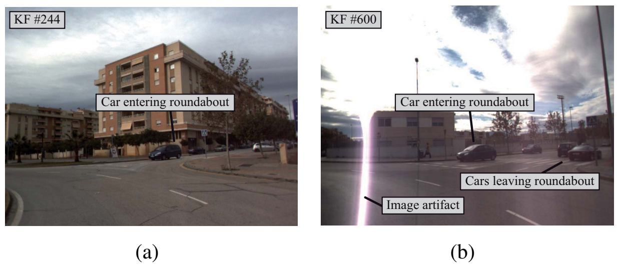

Figure 21. Example of problematic images for the real outdoors dataset. (a) A car entering a roundabout and (b) image artifact and moving cars at a roundabout.

Figure 22. Example image for the KITTI outdoors dataset.

One of the main drawbacks of this dataset stands on the relatively small baseline of the employed stereo camera (∼12 cm), which makes difficult the estimation of distant points' 3D position. Still, the achieved results can be considered to be promising. A comparison between the estimated camera paths and the GPS-based ground truth is shown in Fig. 19 while the errors between both trajectories are presented in Fig. 20. The camera path estimated by the GBA method is only depicted up to keyframe #250.

Moreover, it has to be noted that, in some areas of this dataset, the environment is not static due to the presence of moving cars and/or pedestrians. Since our system works under the assumption of a non-dynamic environment, this leads to large inaccuracies in the camera path estimation. Hence, manual assistance has been required in such areas. In particular, a car approaching the roundabout marked as 'A' in Fig. 19 and the presence of some cars entering and leaving the roundabout marked as 'B' in the same figure (nearly the end of the trajectory) are the specific areas where capturing keypoints on moving objects has been avoided. Example images of these situations are shown in Fig. 21. Finally, the presence of image artifacts due to direct sunlight (refer to Fig. 21(b)) has also been managed in a similar way at the very end of the dataset.

Finally, we have also taken segment 7 from the KITTI's road datasets to test our approach. This segment contains a set of 1100 stereo images of size 1226×370 px. (refer to Fig. 22 for an example image) captured by a car while moving along a ∼0.7 km long trajectory and performing a loop. In this case, the stereo camera baseline is significantly wider than the dataset employed in the previous experiment and, similarly to it, ground truth is available for this dataset.

Again, we have measured the insertion time for new keyframes (including graph optimization) for both the GBA and the SRBA approaches, presenting the results in Fig. 23. Note that, similarly to the previous experiment, the high number of landmarks in the map prevent the process of inserting new keyframes for our GBA implementation to perform in reasonable time. Loop closure detection is

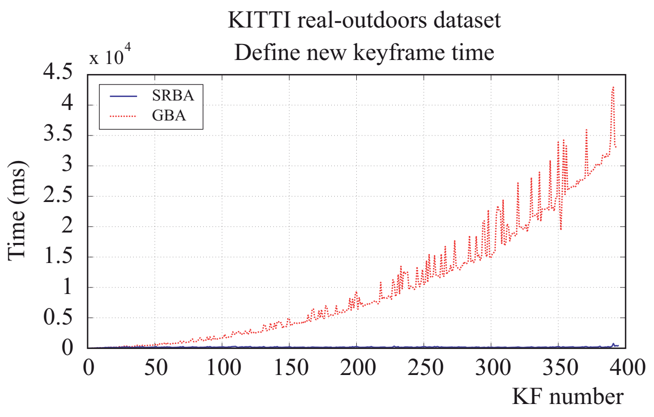

Figure 23. Inserting new keyframe time (including graph optimization) comparison for SRBA (blue-solid line) and GBA (red-dashed line) for the KITTI real outdoors dataset.

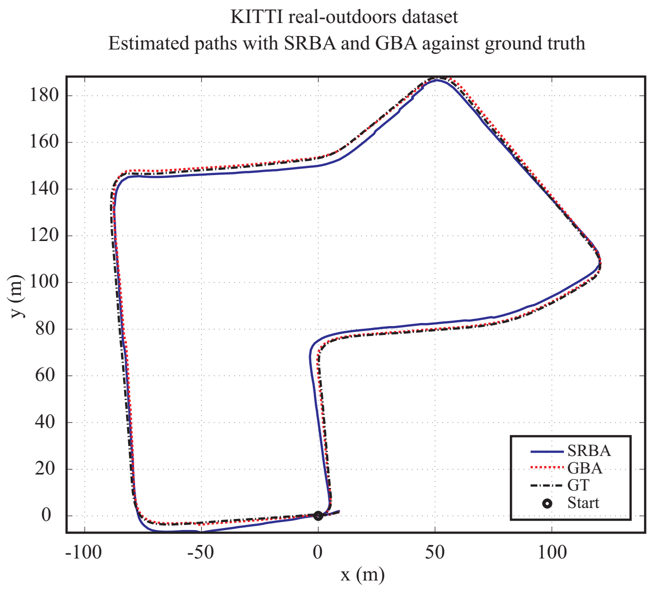

Figure 24. Estimated paths for the KITTI real outdoors dataset with SRBA (blue-solid line) and GBA (red-dashed line). Ground truth is also represented (black dashed-dotted line).

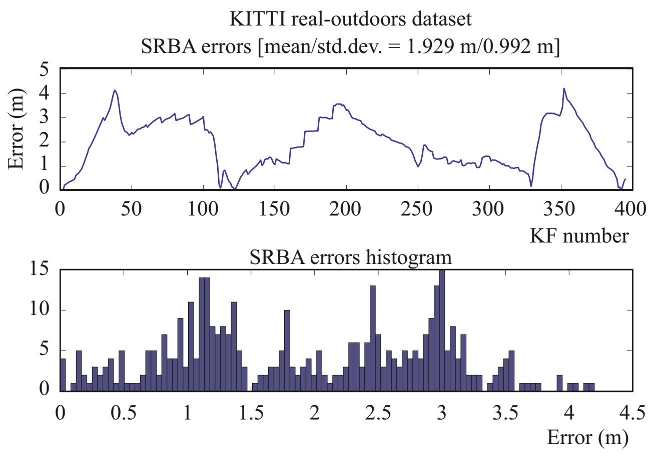

Figure 25. Absolute error and histogram of errors for the SRBA method with respect to the ground truth for the KITTI real outdoors dataset.

achieved near to KF#400 for the SRBA approach, hence the small peak shown in the figure at that point.

The paths estimated by both methods are compared against the provided ground truth as shown in Fig. 24. The graph built contained ∼400 keyframes for the whole trajectory and the resulting map was composed by more than 36k landmarks. Finally, the evolution and the histogram of the errors committed by the SRBA method with respect to the ground truth are presented in Fig. 25. A video showing SRBA operation for this experiment can be found in `https://goo.gl/1Ap4p9`.

7 Conclusion

This paper has addressed the development of a blended relative-global approach to Bundle Adjustment, coined Sparser Relative Bundle Adjustment (SRBA), which allows for a continuum of strategies ranging from classic linear RBA to hybrid submapping with local maps. This flexibility leads to graphs than can be optimized more efficiently than those built as previously reported in the literature, ensuring a bounded-time operation even in the presence of loop closures, regardless their size.

In this work we have extended the preliminary description of SRBA in (Blanco et al. 2013), proven its constant-time nature by providing in-depth listings for the most important algorithms, and validated its performance in the context of stereo visual SLAM. For that, we have developed a complete system combining a front-end that integrates several state-of-the-art computer vision techniques and an SRBA-based back-end that operates as graph optimizer. The presented front-end relies on ORB features as image keypoints and computes camera ego-motion through a visual odometry method which is robust against the presence of outliers. Data association, in turn, is assisted by a bag of words built upon ORB binary descriptors that restricts the search area when looking for loop closures.

A set of experiments in both indoor and outdoor conditions is presented to demonstrate the capabilities of our method when tackling with visual SLAM in comparison with other BA approaches in terms of accuracy and efficiency. Future works include introducing further optimizations in our SRBA implementation, as well as analyzing and introducing more graph maintenance operations to improve its suitability as a long-term SLAM solution.

Acknowledgment

This work has been partially funded by the Spanish *Ministerio de Ciencia e Innovación* under the projects ”TAROTH: New developments toward a Robot at Home” (Contract DPI 2011-25483) and ”PROMOVE: Advances in mobile robotics for promoting independent life of elders” (Contract DPI 2014-55826-R). Both projects are co-founded by *Fondo Europeo de Desarrollo Regional* – FEDER.

References

Agarwal, S., Snavely, N., Seitz, S. M. and Szeliski, R. (2010), Bundle adjustment in the large, *in* ‘European Conference on Computer Vision (ECCV)’, Springer, pp. 29–42.

Anderson, S., MacTavish, K. and Barfoot, T. D. (2015), 'Relative continuous-time slam', *The International Journal of Robotics Research* **34**(12), 1453–1479.

Blanco-Claraco, J.-L. (2013), 'User guide for `libmrpt-srba`: A generic c++ framework for relative bundle adjustment (rba)', http://www.mrpt.org/srba.

Blanco-Claraco, J.-L., Moreno-Dueñas, F.-A. and González-Jiménez, J. (2014), 'The málaga urban dataset: High-rate stereo and lidar in a realistic urban scenario', *The International Journal of Robotics Research* **33**(2), 207–214.

Blanco, J., Fernandez-Madrigal, J. and Gonzalez, J. (2008), 'Towards a Unified Bayesian Approach to Hybrid Metric-Topological SLAM', *IEEE Transactions on Robotics* **24**(2), 259–270.

Blanco, J., González, J. and Fernández-Madrigal, J.-A. (2010), 'Optimal filtering for non-parametric observation models: applications to localization and slam', *The International Journal of Robotics Research* **29**(14), 1726–1742.

Blanco, J.-L. (2010), A tutorial on SE(3) transformation parameterizations and on-manifold optimization, Technical report, University of Malaga.

Blanco, J.-L., González-Jiménez, J. and Fernández-Madrigal, J.-A. (2012), 'An alternative to the mahalanobis distance for determining optimal correspondences in data association', *IEEE Transactions on Robotics (T-RO)* **28**(4), 980–986.

Blanco, J.-L., González-Jiménez, J. and Fernández-Madrigal, J.-A. (2013), Sparser relative bundle adjustment (srba): constant-time maintenance and local optimization of arbitrarily large maps, *in* 'IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 70–77.

Bradski, G. (2000), 'The OpenCV Library', *Dr. Dobb's Journal of Software Tools* .

Calonder, M., Lepetit, V., Strecha, C. and Fua, P. (2010), Brief: Binary robust independent elementary features, *in* 'IEEE European Conference on Computer Vision (ECCV)', Springer, pp. 778–792.

Civera, J., Davison, A. and Montiel, J. (2007), Inverse depth to depth conversion for monocular slam, *in* 'IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 2778–2783.

Concha, A. and Civera, J. (2015), An evaluation of robust cost functions for rgb direct mapping, *in* 'European conference on mobile robotics (ECMR15)'.

Davis, T. A. (2006), *Direct methods for sparse linear systems*, Vol. 2, Society for Industrial and Applied Mathematics (Siam).

Eade, E. and Drummond, T. (2008), Unified loop closing and recovery for real time monocular slam, *in* 'Proceedings of the British Machine Vision Conference', BMVA Press, pp. 6.1–6.10. doi:10.5244/C.22.6.

Fritsch, J., Kuehnl, T. and Geiger, A. (2013), A new performance measure and evaluation benchmark for road detection algorithms, *in* 'International Conference on Intelligent Transportation Systems (ITSC)', IEEE, pp. 1693–1700.

Galvez-Lopez, D. and Tardos, J. D. (2012), 'Bags of binary words for fast place recognition in image sequences', *IEEE Transactions on Robotics* **28**(5), 1188–1197.

Geiger, A., Lenz, P. and Urtasun, R. (2012), Are we ready for autonomous driving? the kitti vision benchmark suite, *in* 'IEEE Conference on Computer Vision and Pattern Recognition (CVPR)', IEEE, pp. 3354–3361.

Gonzalez, J., Galindo, C., Blanco, J., Fernandez-Madrigal, J., Arevalo, V. and Moreno, F. (2009), Sancho, a fair host robot. a description, *in* 'IEEE International Conference on Mechatronics (ICM)', pp. 1–6.

Grisetti, G., Kummerle, R., Stachniss, C., Frese, U. and Hertzberg, C. (2010), Hierarchical optimization on manifolds for online 2d and 3d mapping, *in* 'Robotics and Automation (ICRA), 2010 IEEE International Conference on', IEEE, pp. 273–278.

Hamming, R. W. (1950), 'Error detecting and error correcting codes', *Bell System technical journal* **29**(2), 147–160.

Hartley, R. and Kahl, F. (2007), Optimal algorithms in multiview geometry, *in* Y. Yagi, S. Kang, I. Kweon and H. Zha, eds, 'Computer Vision – ACCV', Vol. 4843 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 13–34.

Huber, P., Ronchetti, E. and MyiLibrary (1981), *Robust statistics*, Vol. 1, Wiley Online Library.

Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J. and Dellaert, F. (2012), 'iSAM2: Incremental smoothing and mapping using the Bayes tree', *Intl. J. of Robotics Research, IJRR* **31**(2), 217–236.

Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J. J. and Dellaert, F. (2011), 'isam2: Incremental smoothing and mapping using the bayes tree', *The International Journal of Robotics Research* **31**(2), 216–235.

Kaess, M., Ranganathan, A. and Dellaert, F. (2008), 'isam: Incremental smoothing and mapping', *IEEE Transactions on Robotics* **24**(6), 1365–1378.

Klein, G. and Murray, D. (2007), Parallel tracking and mapping for small ar workspaces, *in* 'IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR)', IEEE, pp. 225–234.

Konolige, K. and Agrawal, M. (2008), 'Frameslam: From bundle adjustment to real-time visual mapping', *IEEE Transactions on Robotics* **24**(5), 1066–1077.

Lim, J., Frahm, J.-M. and Pollefeys, M. (2011), Online environment mapping, *in* 'IEEE Conference on Computer Vision and Pattern Recognition (CVPR)', IEEE, pp. 3489–3496.

Moore, E. F. (1959), *The shortest path through a maze*, Bell Telephone System.

Moreno, F. A., Blanco, J.-L. and Gonzalez, J. (2009), 'Stereo vision specific models for particle filter-based slam', *Robotics and Autonomous Systems* **57**(9), 955–970.

Moreno, F.-A., Blanco, J.-L. and González-Jiménez, J. (2013), Erode: An efficient and robust outlier detector and its application to stereovisual odometry, *in* 'IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 4691–4697.

Mur-Artal, R., Montiel, J. and Tardos, J. D. (2015), 'Orb-slam: a versatile and accurate monocular slam system', *arXiv preprint arXiv:1502.00956* .

Rosten, E. and Drummond, T. (2006), Machine learning for high-speed corner detection, *in* 'European Conference on Computer Vision (ECCV)', Springer Berlin Heidelberg, pp. 430–443.

Rublee, E., Rabaud, V., Konolige, K. and Bradski, G. (2011), Orb: an efficient alternative to sift or surf, *in* 'IEEE International Conference on Computer Vision (ICCV)', IEEE, pp. 2564–2571.

Sibley, D., Mei, C., Reid, I. and Newman, P. (2009), Adaptive relative bundle adjustment., *in* 'Robotics: science and systems', Vol. 32, MIT Press, pp. 33–40.

Sibley, G. (2009), 'Relative bundle adjustment', *Department of Engineering Science, Oxford University, Tech. Rep* **2307**(09).

Sibley, G., Matthies, L. and Sukhatme, G. (2010), 'Sliding window filter with application to planetary landing', *Journal of Field Robotics* **27**(5), 587–608.

Sibley, G., Mei, C., Reid, I. and Newman, P. (2010), 'Vast-scale outdoor navigation using adaptive relative bundle adjustment', *The International Journal of Robotics Research* **29**(8), 958–980.

Strasdat, H., Montiel, J. and Davison, A. J. (2010), Real-time monocular slam: Why filter?, *in* 'IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 2657–2664.

Strasdat, H., Montiel, J. M. and Davison, A. J. (2012), 'Visual slam: why filter?', *Image and Vision Computing* **30**(2), 65–77.

Thrun, S., Burgard, W. and Fox, D. (2005), *Probabilistic Robotics*, The MIT Press.

Triggs, B., McLauchlan, P. F., Hartley, R. I. and Fitzgibbon, A. W. (2000), Bundle adjustmenta modern synthesis, *in* 'Vision algorithms: theory and practice', Springer, pp. 298–372.

Wolf, J., Burgard, W. and Burkhardt, H. (2002), Robust vision-based localization for mobile robots using an image retrieval system based on invariant features, *in* 'Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)', Vol. 1, IEE, pp. 359–365. doi:10.1109/ROBOT.2002.1013387.