2013 IEEE International Conference on Robotics and Automation (ICRA)
Karlsruhe, Germany, May 6-10, 2013

Sparser Relative Bundle Adjustment (SRBA): constant-time maintenance and local optimization of arbitrarily large maps

José-Luis Blanco[1], Javier González-Jiménez[2] and Juan-Antonio Fernández-Madrigal [3]

Abstract— **In this paper we defend the superior scalability of the Relative Bundle Adjustment (RBA) framework for tackling with the SLAM problem. Although such a statement was already done with the introduction of the sliding window (SW) solution to RBA [16], we claim that the map extension that can be maintained locally consistent for some fixed computational cost critically depends on the specific pattern in which new keyframes are connected to previous ones. By rethinking from scratch what we call *loop closures* in relative coordinates we will show the unexploited flexibility of the RBA framework, which allows us a continuum of strategies from pure relative BA to hybrid submapping with local maps. In this work we derive a systematic way of constructing the problem graph which lies close to submapping and which generates graphs that can be solved more efficiently than those built as previously reported in the literature. As a necessary tool we also present an algorithm for incrementally updating all the spanning-trees demanded by any efficient solution to RBA. Under weak assumptions on the map, and implemented on carefully designed data structures, it is demonstrated to run in bounded time, no matter how large the map becomes. We also present experiments with a synthetic dataset of 55K keyframes in a world of 4.3M landmarks. Our C++ implementation has been released as open source.**

I. INTRODUCTION

Most recent proposals that deal with the simultaneous estimation of a robot path and the unknown location of all observed landmarks use a graph model where nodes are the unknowns (poses and landmarks) and edges represent constraints (e.g. observations, odometry). Robot poses are then typically called *keyframes* (KFs) when working with imaging sensors. Under the assumption of all observation errors being Gaussians it is easily demonstrated [7] that a least-squares minimization of the mismatch between observations and predictions from the estimated model becomes the maximum likelihood estimator for the problem. Typically, sparse algebra approaches are employed to exploit the inherent sparsity of the systems of linear equations that appear during this process. In the computer vision community such methods are dubbed *Bundle Adjustment* (BA) [19], while similar methods in mobile robotics, which may estimate only the robot path or both the path and the landmarks, receive the names of *Graph-SLAM* [9] or *view-based SLAM* [6].

In spite of the great success of this family of techniques, scalability is still an open issue. Even for implementations exploiting sparse solutions, computational complexity may become $O(n^3)$ (with n the number of KFs) if the number of loop closures is proportional to the length of the path [13]. A loop closure is typically understood to be the re-observation of previously seen landmarks that got out of sight during the traversal of a path (the *loop*) excluding going forward and then backwards over the same track. For example, a loop closure occurs in Fig. 1 because landmark a was discovered from KF #0 and then re-observed from KF #5.

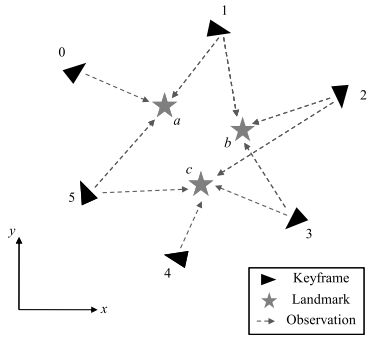


Fig. 1. A miniature SLAM problem as seen in global coordinates. The goal is estimating the positions of keyframes and landmarks from observation data. Notice the loop closure at keyframe #5, where landmark a is redetected.

We call Global Bundle Adjustment (GBA) or global SLAM to all those BA or graph-SLAM methods using only one (thus, global) frame of reference. Under this parameterization, there exists no upper bound for the computational complexity needed to process a loop closure, since the larger the loop, the more KF and landmark coordinates that require a correction.

If we pursue life-long operation of robots it is reasonable to aim at an ideal bounded-time complexity for localization and mapping. We believe that Relative Bundle Adjustment (RBA), a framework similar to graph-SLAM or GBA but where all coordinates are relative, has the potential to take us closer to this ideal.

In RBA, unknowns are now the relative poses between KFs and the position of landmarks with respect to (wrt) some KF which we will call the *base* KF for that landmark –commonly, the KF from which it was firstly observed. Fig. 2 shows a model of the previous example under this perspective.

The sliding window (SW-RBA) solution to RBA [16] achieves a constant time computational cost by only optimizing the unknowns up to a certain topological distance (the "window") from the latest KF inserted in the map, i.e. the "current" pose. An adaptive size for this window was also investigated [17] (ASW-RBA), but in any case

[1]J.L. Blanco is with the Department of Mechanical Engineering, University of Almería, Spain joseluisblancoc@gmail.com
[2],[3]J. González-Jiménez and J.A. Fernández-Madrigal are with the Department of System Engineering and Automation, University of Málaga, Spain javiergonzalez@uma.es, jafma@ctima.uma.es

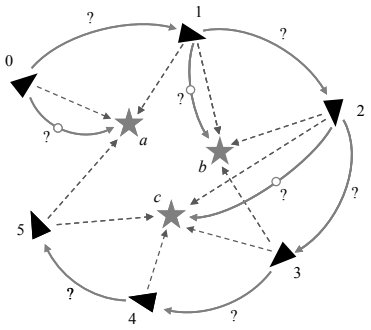


Fig. 2. One possible reformulation of the problem in Fig. 1 as RBA, where unknowns are now the thick edges with a question mark. Unknown keyframe-to-landmark edges, marked with a circle mid-mark, connect landmarks with the keyframes from which they are observed for the first time. Two central topics to this paper are: (i) is this a loop closure for RBA? and (ii) is this the most *efficient* way to parameterize the problem?

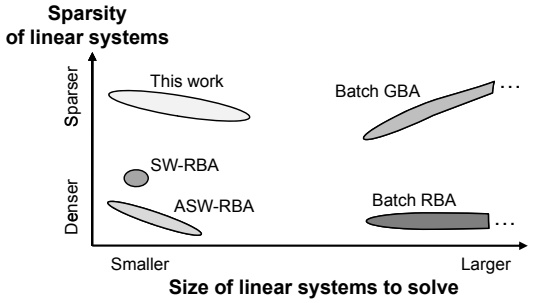


Fig. 3. A qualitative classification of SLAM methods under the criterion of sparsity vs. size of the linear systems that must be solved. Represented methods include batch Global Bundle Adjustment (GBA) [19], Batch and sliding window (SW) solutions to Relative Bundle Adjustment (RBA) [16] and its Adaptive sliding window (ASW) version [17], and this work.

it remains independent of the map size. Thus, the RBA framework provides an efficient approximate solution to SLAM, estimating locally consistent maps.

In RBA it seems natural to create relative-coordinate edges from each KF to its preceding one, thus KFs tend to form a *linear graph* as in Fig. 2. Loop closures are the obvious exceptions at where to build something different than a linear graph. However, loop closures in RBA introduce a number of new problems without parallel in other SLAM frameworks which have not been properly studied yet in the literature. In global SLAM, loop closures do not modify the problem unknowns, only the constraints between them. In contrast, in RBA they may or may not introduce new unknowns (new relative poses between KFs): we can decide what is more convenient under some criterion. The problem in Fig. 2 is a loop closure in the common sense, but is not for RBA since KFs still form an acyclic graph. An RBA loop closure occurs when a new KF is added to the graph with more than two new edges. In the figure, one possibility for the loop closure in KF #5 would be adding an extra edge $\#5 \rightarrow \#0$. This paper will discuss the pros and cons of adding such edges in RBA.

Furthermore, we can go one step forward and ask ourselves if connecting all poses in a linear graph is really *optimal* in some sense, or if there are other alternatives. In this paper we claim that RBA offers an unprecedented flexibility in this aspect in comparison to other SLAM frameworks and that, indeed, some connection patterns lead to systems of equations that are much more efficient to solve than others. In particular, we will see how relying on linear graphs introduces large dense blocks in the Hessian matrix of the problem, rendering the computational cost of the estimation higher than necesary. Fig. 3 qualitatively compares our proposed connection pattern to other alternatives.

Finally, but of the most practical importance, we address the problem of maintaining the shortest-path spanning trees (STs) required by RBA in an incremental way for arbitrarily-complex loop closures (the relation between STs and RBA is clarified in section II-B). Under mild assumptions, it is shown that the method runs in constant-time wrt the size of the map.

In the following we will assume that the problem of data association, finding the corresponding map landmark for each observation, is solved separately [15], [4]. Thus, data association clearly becomes the present bottleneck towards achieving a complete constant-time solution to SLAM.

The source code of our implementation has been released as the C++ library `libmrpt-srba`[1].

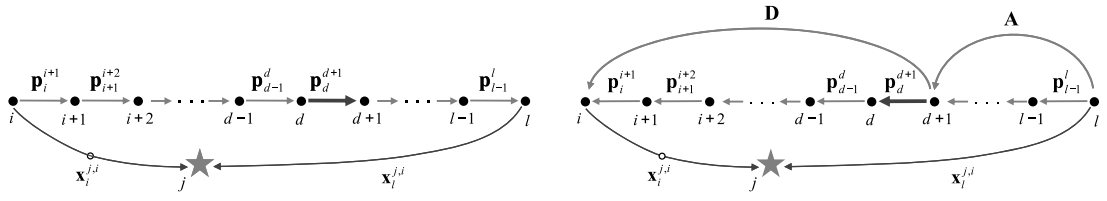
II. Problem Statement

A. Graphical representation and Notation

In global SLAM there exists an immediate correspondence between the graphical representation of (i) keyframes (poses), landmarks and their constraints (observations), and (ii) the graphical models (Dynamic Bayesian Networks or Markov Networks) used to drive the Bayesian or least-squares estimation. For RBA this connection is not as straightforward since the unknowns (nodes in graphical models) are represented as edges. Although it is possible to convert an RBA graph into its corresponding graphical model, in this work only the former will be represented in order to better preserve the physical meaning of the variables.

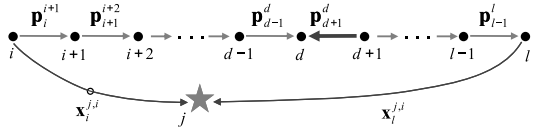
Thus, an RBA graph consists of a *directed* graph, whose nodes are either KFs or landmarks. Edges between them represent either: (i) observations, (ii) relative poses of one KF wrt another (a KF-to-KF edge), or (iii) the relative position of a landmark wrt its *base KF* (a KF-to-LM edge). The two latter kinds of edges are unknowns to be determined from the observations. Thus, notice how graphs can contain two fundamentally different types of edges between KFs and landmarks: observations (known data) and KF-to-LM edges (unknowns). When represented graphically we will denote the latter as edges with a circular mid-mark in order to help telling ones from the others. Relative poses and positions can be parameterized in a number of ways, but since this becomes irrelevant for the present work we will assume that poses are represented by a translation plus three Euler angles and positions by their Cartesian coordinates.

[1]Available online: `http://www.mrpt.org/srba`

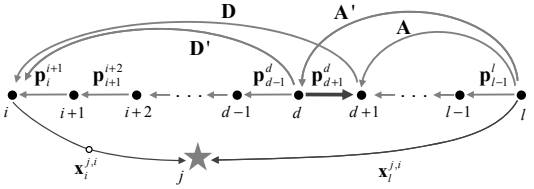


(a) Spanning tree for $i \rightarrow l$, with a direct edge $d \rightarrow d+1$

(b) Chain of poses $i \leftarrow l$, with a direct edge $d \rightarrow d+1$



(c) Spanning tree for $i \rightarrow l$, with an inverted edge $d+1 \rightarrow d$



(d) Chain of poses $i \leftarrow l$, with an inverted edge $d+1 \rightarrow d$

Fig. 4. Notation used for the derivation of the required Jacobians with respect to a relative pose $\mathbf{p}_d^{d+1}$ (or $\mathbf{p}_{d+1}^{d}$) in the middle of a spanning tree between a keyframe l (observing a landmark j) and the keyframe i (the root of the tree) used as local coordinate base for that landmark.

About the direction of KF-to-KF edges, that is, whether an edge is represented as $a \rightarrow b$ or $b \rightarrow a$, we must remark that the choice to pick one of them when introducing a new edge is completely arbitrary. This is justified on the basis that for the sake of determining STs, the graph of KFs is always treated as if it was *undirected*. For convention we will always choose as origin the KF with the lowest identifier number. Furthermore, for convenience in the formulation below we assume that the unknown relative pose $\mathbf{p}_a^b$ for an edge $a \rightarrow b$, with $a < b$, actually stands for the *inverse pose*; that is, $\mathbf{p}_a^b$ is the pose of the KF a as seen from the frame of coordinates of KF b.

This must be kept in mind while addressing chains of poses along STs, undoubtedly the most important construction that appears while solving RBA (see section II-B). For each KF we need the ST composed of the shortest paths to all other KFs from that root KF, and up to some predefined topological distance D_{max}. Note that the direction of KF-to-KF edges and the existence of observations or KF-to-LM edges are all ignored during the construction of these trees, i.e. as if we had an *undirected* graph of KFs. An algorithm to build those trees is presented in section IV.

The notation of the involved variables is depicted with Fig. 4. As can be observed in the example of Fig. 4(a)–(b), relative poses are defined in the opposite direction than directed edges, where the former represents KF-to-KF edges and the latter the geometrical meaning of the relative poses. Paths along these trees are the shortest ones from the root KF, thus the figure represents the shortest sequence of edges between two KFs i and l. We will denote arbitrary relative poses along such a path as $\mathbf{p}_d^{d+1}$ when the edge goes in the direction $i \rightarrow l$ (see Fig. 4(a)–(b)) or as $\mathbf{p}_{d+1}^{d}$ when heading in the opposite direction (see Fig. 4(c)–(d)).

The relative position of the j–th landmark wrt the KF o will be denoted as $\mathbf{x}_o^{j,b}$, with b the base KF of the landmark (unique, does not vary over time). Therefore, the $\mathbf{x}_b^{j,b}$ will always be unknowns (KF-to-LM edges) while $\mathbf{x}_o^{j,b}$ with $b \neq o$ are relative positions (a function of unknowns) involved in observations gathered from KF o.

Other unexploited possibilities in RBA are the definition of landmarks with a fixed (known) relative position (useful, for example, when using a variety of sensors and one of them provides accurate range information) and "virtual" fixed landmarks which can model constraints between KFs (e.g. odometry). We do not cover them in this work.

B. Least-squares estimator

Given a set of N_o observations $\{\mathbf{z}_i\}$ corrupted with additive zero-mean Gaussian noise of known information (inverse covariance) matrices $\mathbf{\Lambda}_i$, the Maximum Likelihood Estimation of the unknowns (all relative poses $\mathbf{p}$ and all relative positions $\mathbf{x}$) becomes the minimization of the cost function [7]:

$$\begin{aligned} F(\mathbf{p},\mathbf{x}) &= \frac{1}{2}\sum_{i=1}^{N_o} \mathbf{\Delta z}_i^\top \mathbf{\Lambda}_i \mathbf{\Delta z}_i \\ \mathbf{\Delta z}_i &= \mathbf{h}_i(\mathbf{p},\mathbf{x}) - \mathbf{z}_i \end{aligned} \tag{1}$$

with $\mathbf{h}(\cdot)$ being the observation model. By means of a first-order Taylor series expansion of the latter and using the Gauss-Newton approximation of $F(\cdot)$'s Hessian, $\mathbf{H} \approx \mathbf{J}^\top \mathbf{\Lambda} \mathbf{J}$ with $\mathbf{J} = \frac{\partial \mathbf{h}(\mathbf{p},\mathbf{x})}{\partial \{\mathbf{p},\mathbf{x}\}}$, we can solve for the increments in the unknowns $[\mathbf{\Delta p}\ \mathbf{\Delta x}]^\top$ that successively approach the most likely estimation:

$$\left(\mathbf{J}^\top \mathbf{\Lambda} \mathbf{J}\right) \begin{pmatrix} \mathbf{\Delta p} \\ \mathbf{\Delta x} \end{pmatrix} = -\mathbf{J}^\top \mathbf{\Lambda} \mathbf{\Delta z} \tag{2}$$

When the initial estimation is far from the real solution it becomes a better idea to employ the Levenberg-Marquardt (implemented in this work) or the Powell's dog leg algorithms instead [12]. Note that the uncertainty of the estimation is available in the form of the Hessian, which is proportional to the estimation information matrix. Efficiently recovering the cross-covariances between the different landmarks and poses for data association (e.g. via JCBB [15]) is a cumbersome problem explored elsewhere [8], [10].

The fundamental basis for any efficient solution to this optimization problem is exploiting the sparse structure of the Jacobian $\mathbf{J}$ [19]. Since each observation depends on only a few unknowns, most entries in the Jacobian are typically zeros (see the example of Fig. 5(b)). We find here an essential difference between RBA and global SLAM: while in the latter the number of nonzero entries only depends on the number of observations, in RBA it is also strongly determined by the pattern in which KF-to-KF edges are connected –see section III. Our implementation accounts for the sparsity of the problem structure to efficiently store and compute only the required submatrices, as described in section II-C.

Each observation $\mathbf{z}_i$ contributes to the cost function by means of a row of block matrices in the sparse Jacobian $\mathbf{J}$. Observations are functions $\mathbf{h}(\cdot)$ of the relative position of landmarks wrt an *observer KF* l (i.e. the KF from which an observation is taken), while our map unknowns are the coordinates of a landmark j referred to its base KF i, thus we arrive at the model $\mathbf{h}(\mathbf{x}_l^{j,i})$ – see notation above. The observer l and the base i are typically different KFs, thus we need their relative pose in order to evaluate $\mathbf{x}_l^{j,i}$. In general we can find the relative pose of any pair of KFs by composing a chain of poses for all the KF-to-KF edges along the topological path that takes from l to i – refer to Fig. 4. The minimum number of variables involved is found by following the shortest of such paths. Here is where we find the utility of maintaining STs rooted at each KF and containing the shortest path to any other KF around.

The Jacobian wrt the unknown KF-to-LM edge $\mathbf{x}_i^{j,i}$ is easily found with the chain rule of derivatives. Obtaining the Jacobian blocks wrt any KF-to-KF edge in the shortest path, say $\mathbf{p}_d^{d+1}$ in Fig. 4(a)–(b), follows from firstly applying the chain rule:

$$\frac{\partial \mathbf{h}(\mathbf{x}_l^{i,j})}{\partial \mathbf{p}_d^{d+1}} = \left.\frac{\partial \mathbf{h}(\mathbf{x}')}{\partial \mathbf{x}'}\right|_{\mathbf{x}'=\mathbf{x}_l^{i,j}} \left.\frac{\partial \mathbf{x}_l^{i,j}}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}_d^{d+1}} \quad (3)$$

where the first term is trivial to evaluate. The second is better replaced by taking derivatives wrt the more convenient linearization on the pose ($\mathbf{p}$) manifold (ε). By also introducing the chain of poses defined by the shortest path:

$$\left.\frac{\partial \mathbf{x}_l^{i,j}}{\partial \varepsilon}\right|_{\varepsilon=\mathbf{0}} = \left.\frac{\partial(\overbrace{\mathbf{p}_{l-1}^{l}\cdots\mathbf{p}_{d+1}^{d+2}}^{\mathbf{A}}\, e^{\varepsilon}\, \overbrace{\mathbf{p}_{d}^{d+1}\cdots\mathbf{p}_{i}^{i+1}}^{\mathbf{D}}\mathbf{x}_i^{i,j})}{\partial \varepsilon}\right|_{\varepsilon=\mathbf{0}} \quad (4)$$

which has a known expression [1]. The mathematical details on optimization on the SE(3) manifold and the derivation of related Jacobians were described in [1], [7].

Finally, in contrast to previous literature on RBA, we will create graphs with an arbitrary topology, thus the shortest paths from STs will not necessarily have all KF-to-KF edges pointing towards l as in Fig. 4(a). When an intermediate edge $\mathbf{p}_{d+1}^{d}$ is found in the opposite direction, as in Fig. 4(c), derivatives wrt $\mathbf{p}_{d+1}^{d}$ (see Fig. 4(d)) can be shown to lead to:

$$\left.\frac{\partial \mathbf{x}_l^{i,j}}{\partial \varepsilon}\right|_{\varepsilon=\mathbf{0}} = -\left.\frac{\partial(\mathbf{A}' e^{\varepsilon}\, \mathbf{D}' \mathbf{x}_i^{i,j})}{\partial \varepsilon}\right|_{\varepsilon=\mathbf{0}} \quad (5)$$

with $\mathbf{A}' = \mathbf{A}\left(\mathbf{p}_d^{d+1}\right)^{-1}$ and $\mathbf{D}' = \left(\mathbf{p}_d^{d+1}\right)^{-1}\mathbf{D}$.

C. Symbolic constructions

We briefly review the data structures we have designed to exploit the sparsity of the problem.

1) Jacobians: We adopted a column-indexed structure of sparse columns for storing the nonzero Jacobian blocks, an idea already proposed in Spa2D [11]. The list of columns is kept in a non-associative container, with $O(1)$ access and growth costs (C++ STL's `std::deque`), while each column maintains its sparse entries in an associative container (`std::map`). The unique two operations on the latter are: (i) inserting new rows (observations), doable in $O(1)$ instead of the generic logarithmic cost since we know the insertion point, and (ii) looking for the intersection set with another column while building the Hessian. Assuming that each landmark is only observed from a bounded number of KFs (i.e. redundant KFs are not indefinitely inserted in the map) this cost does not grow with the map size.

2) Hessian: For any subset of unknowns we build and store the sequence of matrix operations required to construct each of the three blocks of the Hessian "primary structure" [19]: the Hessian for poses $\mathbf{H_p}$, for landmarks $\mathbf{H_x}$ and the cross terms $\mathbf{H_{px}}$. By storing references (pointers) to the placeholders where Jacobians will be numerically evaluated, the update of the Hessian becomes a purely mechanical reproduction of the stored program without any further logic.

3) Schür reduced system: Taking into account the sparsity of the $\mathbf{H_x}$ block (the "secondary structure") provides a huge computational advantage. Building the Schür poses-only reduced system is approached by means of what we call *symbolic Schür structure*, a storage of the sequence of matrix operations and numeric matrix placeholders required for its construction, such that during optimization steps no further logic is required.

Curiously, as a result of our custom data structures handling all the problem sparsity and the limited number of unknowns found in RBA, there is not much sparsity to be exploited when solving the final system of equations $\mathbf{Ax} = \mathbf{b}$. Therefore, and contrarily to most recent global SLAM methods, solving the linear equations is more efficient in our case by applying dense Cholesky algorithms instead of the symbolic (sparse) versions [5].

4) Spanning trees: Trees are maintained only up to a desired maximum depth D_{max}, which also settles the maximum map area which is assured to be maintained locally consistent (see section III). We store STs as a symbolic structure comprising two *tables*: $\mathcal{ST}.\mathcal{D}[i][j]$ and $\mathcal{ST}.\mathcal{N}[i][j]$. For any pair of KFs i and j, the former keeps the length of the shortest path (distance) from i to j while the latter indicates the next KF found in the direction of that path.

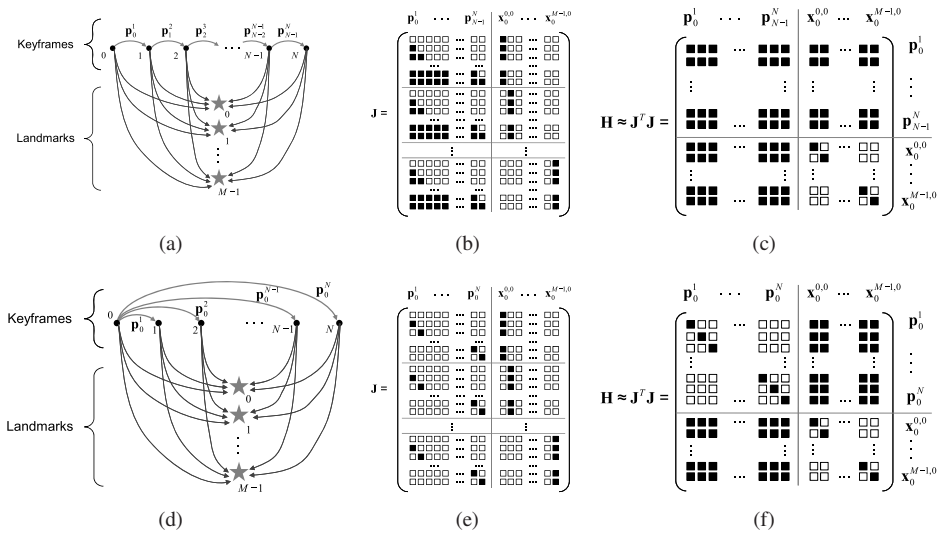


Fig. 5. An example of how two parameterizations of the same problem (i.e. (a)–(c) relative BA, (d)–(f) global BA) with the same number of unknown KF-to-KF edges affect the sparsity of the Hessian $\mathbf{H}$. Each observation requiring a chain of relative poses of length L requires the evaluation of $L+1$ Jacobian blocks, L^2+L+1 matrix multiplications for the Hessian and leads to an $L \times L$ dense block in the same matrix.

Algorithm 1 `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 α_n^i
Output: The updated, locally consistent map

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

// Apply edge-creation policy to decide how to handle loop closures, etc.
while $\varnothing \neq [(i_k \leftrightarrow n) =$ `decide_edge_to_create()`$]$ **do** ▷ $O(N_o \log N_R)$
  `add_kf2kf_edge`$(i_k \leftrightarrow n)$ ▷ Update KF-to-KF edge structures – $O(1)$
  `update_sym_spanning_trees`$(i_k \leftrightarrow n)$ ▷ $O(N_R^2 \log N_R)$
end while ▷ Typ. iterations: $O(\gamma)$

// Update symbolic Jacobian structures ▷ $O\left(N_o(D_{max} + \log N_R)\right)$
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
  `add_observation(` $\underbrace{\mathbf{z}_n^i}_{\text{obs. data}}$, $\underbrace{n}_{\text{observing KF}}$, $\underbrace{\alpha_n^i}_{\text{landmark ID}}$ `)` ▷ $O(D_{max} + \log N_R)$
end for

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

Only the former is assured to be symmetric. Both tables are physically stored together in one row-indexed list of sparse rows. Assuming a bounded degree for the graph of KFs, i.e. that we avoid indefinitely adding new edges to any KF, the number of reachable nodes (N_R) from any ST with a depth D_{max} is also bounded, thus searches and insertions in sparse rows have a bounded complexity of $O(\log N_R)$.

Given any pair of KFs at a maximum topological distance of D_{max}, it is possible to evaluate their numeric relative pose (denoted as $\mathcal{ST}_{num}[i][j]$) in $O(D_{max} \log N_R)$ from these tables.

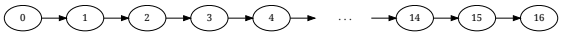
III. Sparser RBA (SRBA)

By changing our policy about how to introduce new edges, RBA can be turned into one of different frameworks:

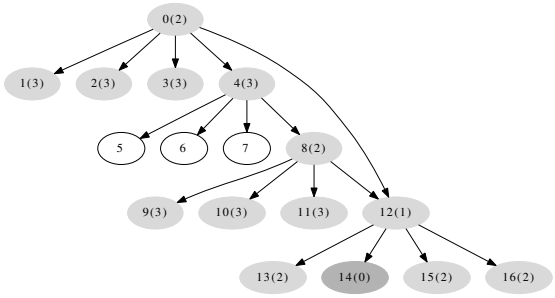
- Every KF is connected to its immediately preceding KF (except for loop closures where extra edges appear) and all KF-to-LM edges' bases are the KF from where each landmark was first observed. This is what we call the "intuitive" choice for RBA, and was the proposed approach in its introduction ([16], [17]).
- A single KF appears as the base for all edges (KF-to-KF and KF-to-LM): Under this situation RBA becomes exactly the same problem as global BA (GBA).
- Different KFs are dynamically selected as the "local origin" of coordinates over time. The present work advances in this direction, which may be seen as a "blended solution" in between the two previous ones.

Thus, RBA is capable of a seamless integration of benefits from both pure relative and global coordinates: from the former we have the possibility of closing arbitrarily-large loops with a bounded computational cost, while from the latter we will incorporate its *sparser* structure (hence the name, SRBA) which for a same number of unknowns is more efficient to solve. This point is illustrated with the example in Fig. 5, where both the Jacobian and the Hessian are clearly denser in RBA than in GBA (for the specific KF-to-KF connection pattern shown here). This follows from the insight that, in RBA, each observation introduces an $L \times L$ dense block in the Hessian with L the shortest path length to the landmark's base KF. Thus one of our goals when adding new edges will be *minimizing the distance between observer and base KFs*.

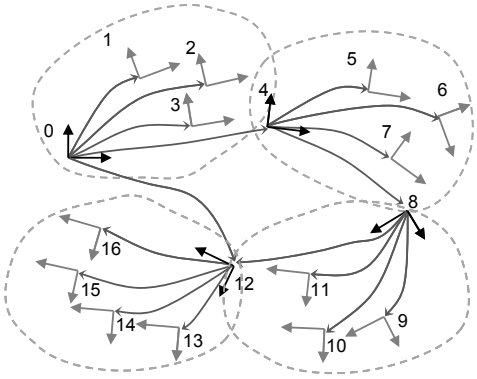
To quantitatively evaluate the cost of each SLAM timestep we summarize our generic SRBA method in Algorithm 1. With the exception of the ST maintenance which is addressed in section IV, space limits do not permit an in-depth analysis of each subalgorithm. The resulting worst case computational complexities are summarized along with the pseudocode, where D_{max} is the maximum depth of the maintained STs,



(a) Linear graph for the "intuitive" RBA policy



(b) The proposed hierarchical-like graph



(c) Geometrical representation

Fig. 6. The "intuitive" edge creation policy for RBA leads to linear graphs of KFs as in (a). Our alternative lay out is shown in (b) for a physical robot path like that in (c). As an example, all the KFs reachable from KF $\#14$ for a $D_{max} = 3$ are shaded in (b) with the distance in parentheses.

N_R the number of reachable nodes in each ST (which is bounded as long as it is the degree of the graph) and N_o the number of observations. A key issue here is modeling the number of edges (not nodes) in a ST. Assuming a constant ratio γ between edges and nodes, we have $O(\gamma N_R)$ edges to optimize in each time step (line 11 in the algorithm).

To sum up, we arrive at three conflicting desiderata:

1) Having a large N_R such that a large portion of the map is kept locally consistent.
2) Keeping γN_R as reduced as possible to reduce the number of unknowns and the cost of solving the associated linear system. This recommends keeping a sparsely connected graph (low γ values).
3) Shortening all paths between observer and base KFs as much as possible to reduce the above-mentioned $O(L^2)$ cost and arrive at sparser Hessians. Shorter paths require a denser graph, increasing γ and conflicting with the previous point.

Our proposal is to find a trade-off by means of an *edge creation policy* which either assures the maximization of some model for the estimated computational cost or relies on heuristic rules. Determining such rules is complex enough for deserving dedicate future research.

In this paper we investigate one heuristic rule, which turns RBA in something similar to hybrid SLAM with submapping [2], [3]. We propose dividing the sequence of KFs into "submaps", such that one KF plays the role of local frame of reference for all other KFs in the submap. As a first approximation, a new submap can be started after the last one fills up to a fixed number of KFs. Whereas the "intuitive" policy in RBA leads to linear graphs, ours generates a hierarchical-like lay out – compare Figs. 6(a)–(b).

All KFs within a submap have one and only one KF-to-KF edge connecting it to its local frame of reference. Whenever a new submap is created, its first KF becomes its local reference and we consider the creation of a number of multiple edges to other submaps, but only between KFs that play the role of local references. With this simple heuristic we assure that most KFs in the graph have a degree of exactly 1, while a large number of potential observations' base KFs are still available at a short topological distance. Consider the example of the KF #14 in the figures, whose ST covers all the nodes shaded in Fig. 6(b) for a D_{max} as reduced as 3 (compare this to the number of reachable KFs in the linear graph of the original SW-RBA, which is typically $2D_{max}$). In practice, $D_{max} = 4$ should be enough since it allows observing landmarks whose bases belong to all neighbor and neighbor of neighbor submaps. Additional edges between reference KFs (e.g. $\#0 \rightarrow \#12$ in the example) represent physical loop closures, as illustrated in Fig. 6(c).

IV. Updateable shortest-path spanning trees

A. The basic idea

The shortest-path STs for all the nodes of a graph $G = (N, E)$ with $|N|$ nodes and $|E|$ edges can be easily built with the classic breadth-first-search (BFS) algorithm [14] in $O\left(|N|\left(|N| + |E|\right)\right)$. Though, we are not interested in the construction of such trees from scratch for an existing graph. Instead, we pursue the simultaneous, incremental construction of both, the graph and all its associated STs. To the best of the authors' knowledge, this problem has not been addressed in the SLAM community before.

Spira and Pan found in 1975 that updating a ST after the addition of a new node to a graph has a worst-case complexity of $O(|N|)$ [18, see theorem 3.3]. It would then seem unfeasible to obtain a bounded execution time algorithm to update all the STs required in RBA for unbounded maps, i.e. for unbounded $|N|$. However, we should highlight that the worst case complexity is related to the maximum topological distance (along existing STs) between the different nodes at which the newest KF is connected to the graph. Since our STs have a maximum depth we therefore avoid the appearance of the total graph size ($|N|$) in the task of updating STs. We describe next the proposed algorithm.

B. The algorithm

We start with an existing graph of KFs, for which we know the shortest path STs of maximum depth D_{max} for all its nodes. This prerequisite is not limiting since we can start with an empty graph and build all those STs incrementally.

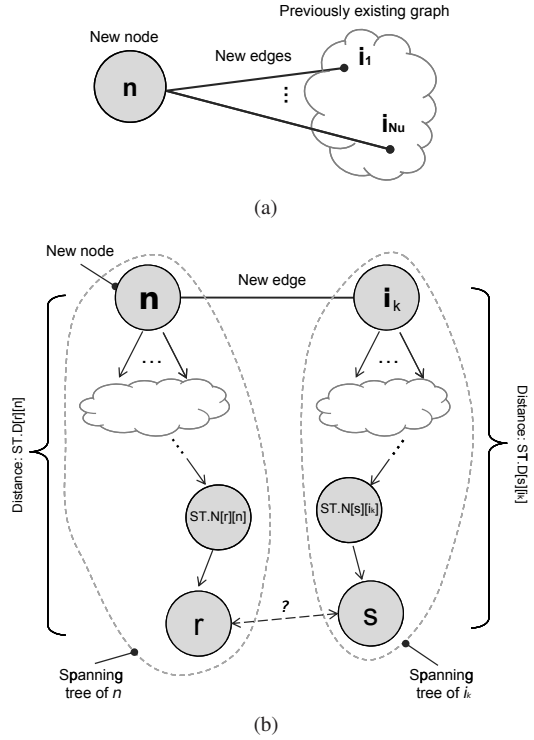


Fig. 7. Notation and basic concepts involved in our updateable shortest-path spanning-tree algorithm.

A new node n is then added to the graph with N_u undirected edges connecting it to nodes i_k, $k = 1...N_u$ – see Fig. 7(a). If only one edge is added the algorithm greatly simplifies but for the sake of generality we will describe our generic procedure which works for any number of new edges N_u.

New edges must be processed one by one. For each new edge to i_k we have the situation depicted in Fig. 7(b): all nodes belonging to the ST of n (including its root n) are now potentially close to all those in the ST of i_k (including the root i_k as well). The distance between any such node r in the ST of n and another s in the ST of i_k is easily evaluated by summing three components: the distance of the way up from r to n ($d_1 = \mathcal{ST}.\mathcal{D}[n][r]$), the new edge $n-i_k$ ($d_2 = 1$) and the way down within the ST of i_k ($d_3 = \mathcal{ST}.\mathcal{D}[i_k][s]$). If nodes r and s were previously at a distance larger than D_{max} (which is reflected by they not being in the current ST of each other) and the resulting distance $d = d_1 + d_2 + d_3$ is D_{max} or less, then both STs must be updated to reflect this new path. If a previous path existed between r and s it must be updated only if the new path is shorter than the currently stored shortest path. Otherwise, the new edge is irrelevant for what respect the shortest path between these nodes. This evaluation is repeated for all potential nodes s and r.

The procedure is summarized in Algorithm 2. 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 fixed D_{max} and 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.

Algorithm 2 `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: $ST_{D_{max}-1}(i_k) \leftarrow \{\forall v / d(v, i_k) \leq D_{max} - 1\}$ ▷ $O(N_R)$
2: $ST_{D_{max}}(n) \leftarrow \{\forall v / d(v, n) \leq D_{max}\}$ ▷ $O(N_R)$
3: **for each** $r \in ST_{D_{max}}(n)$ **do** ▷ $O(N_R)$ iterations
4: **for each** $s \in ST_{D_{max}-1}(i_k)$ **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** $(s \in$ `spanning_tree`(r) **and** $d < \mathcal{ST}.\mathcal{D}[r][s])$ **or** $(s \notin$ `spanning_tree`(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**

V. Experiments and conclusions

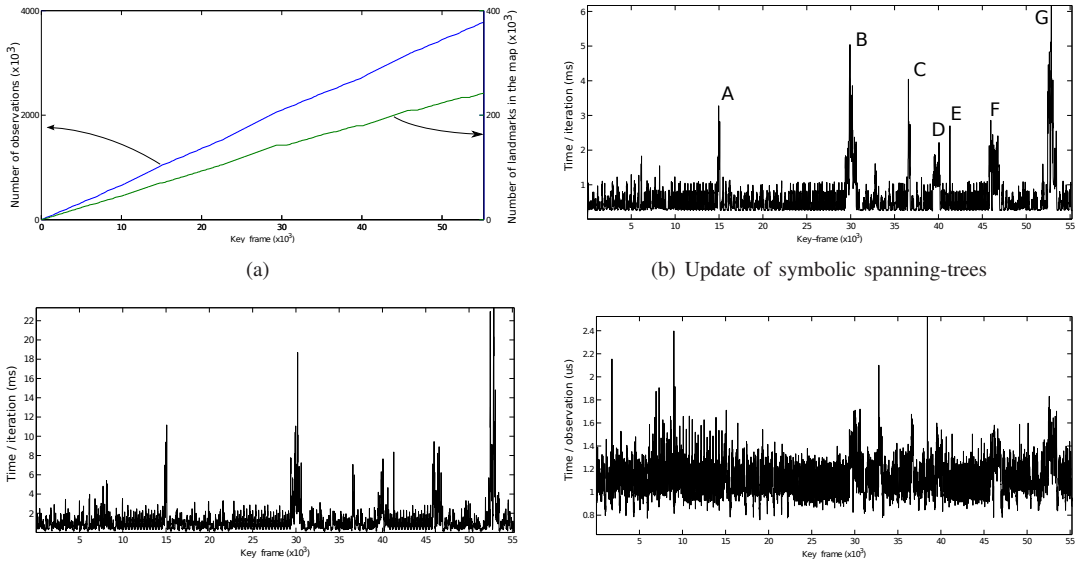
In order to validate the alleged constant-time complexity of our implementation we firstly generated[2] the large synthetic dataset shown in Fig. 9, and comprising 55K keyframes with monocular camera observations. A video is also available online[3]. Then, all observations are processed sequentially while monitoring the execution times of all the relevant stages of the algorithm. Fig. 8(b) shows the results for our ST algorithm, which exhibits the expected constant time complexity (its cost does not grow appreciably during the 55K keyframes) excepting the peaks at loop closures where N_R temporarily grows. In fact, these peaks are independent of the loop length (e.g. the largest loop is closed at "B", but "G" has a higher cost) and can be greatly further reduced by avoiding adding redundant KFs, which is not addressed yet in our implementation.

We also show in Fig. 9(c) one of the most complex stages, building the symbolic Hessian, whose complexity clearly also depends on N_R but not on the size of the map. In contrast, updating the sparse Jacobian structures with new observations is a completely constant-time operation, as seen in Fig. 9(d).

Therefore, we have demonstrated the functionality of our constant-time implementation of an algorithm for maintaining all the STs of an RBA graph, together with a submapping-like policy for edge creation. The latter is, however, just one possibility and further research is required to fully exploit the potential of the SRBA framework and to compare the accuracy of RBA solutions to classic GBA.

[2]The tool designed to generate large simulated datasets has been made available as open source at:
`http://code.google.com/p/recursive-world-toolkit/`

[3]Playlist on YouTube: `http://goo.gl/kP7IS`



(a)

(b) Update of symbolic spanning-trees

(c) Symbolic Hessian construction

(d) Update of symbolic sparse Jacobians

Fig. 8. (a) Number of processed observations and landmarks over time (time indices are KFs). (b) Computational cost (in milliseconds per new KF) of running the proposed algorithm for ST maintenance. Letters correspond to the robot path locations highlighted in Fig. 9. (c) Cost (in milliseconds) of building our symbolic Hessian from sparse Jacobian structures. (d) Time spent on growing our sparse Jacobian data structures (in microseconds per observation). All times are for a single-threaded program running on an Intel i5 2.9GHz.

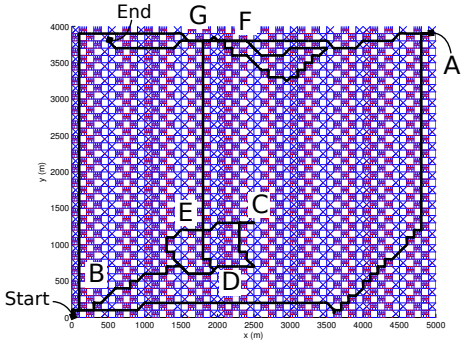


Fig. 9. The test dataset *world_4.3M_3loops* consists of a $55 \cdot 10^3$ KFs path comprising three long loop closures. A total of $3.78 \cdot 10^6$ observations are generated from a synthetic world with $4.37 \cdot 10^6$ landmarks. Thin blue lines are corridor-like pathways where landmarks (not shown here) concentrate, while the thick black line is the path described by the robot.

REFERENCES

[1] J.-L. Blanco, "A tutorial on se(3) transformation parameterizations and on-manifold optimization," University of Malaga, Tech. Rep., Sept. 2010.

[2] J.-L. Blanco, J.-A. Fernández-Madrigal, and J. González-Jiménez, "Towards a unified bayesian approach to hybrid metric-topological slam," *IEEE Transactions on Robotics*, vol. 24, no. 2, pp. 259–270, 2008.

[3] M. Bosse, P. Newman, J. Leonard, and S. Teller, "Simultaneous localization and map building in large-scale cyclic environments using the atlas framework," *The International Journal of Robotics Research*, vol. 23, no. 12, pp. 1113–1139, 2004.

[4] M. Cummins and P. Newman, "Appearance-only slam at large scale with fab-map 2.0," *The International Journal of Robotics Research*, vol. 30, no. 9, pp. 1100–1123, 2011.

[5] T. Davis, *Direct methods for sparse linear systems*. Society for Industrial Mathematics, 2006, vol. 2.

[6] R. Eustice, H. Singh, and J. Leonard, "Exactly sparse delayed-state filters for view-based slam," *IEEE Transactions on Robotics*, vol. 22, no. 6, pp. 1100–1114, 2006.

[7] J.-A. Fernández-Madrigal and J.-L. Blanco, *Simultaneous Localization and Mapping for Mobile Robots: Introduction and Methods*. IGI Global, sep 2012. [Online]. Available: http://www.igi-global.com/book/simultaneous-localization-mapping-mobile-robots/66380

[8] G. Golub and R. Plemmons, "Large-scale geodetic least-squares adjustment by dissection and orthogonal decomposition," *Linear Algebra and Its Applications*, vol. 34, pp. 3–28, 1980.

[9] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *Intelligent Transportation Systems Magazine, IEEE*, vol. 2, no. 4, pp. 31–43, 2010.

[10] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, "isam2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 3281–3288.

[11] K. Konolige, G. Grisetti, R. Kummerle, W. Burgard, B. Limketkai, and R. Vincent, "Efficient sparse pose adjustment for 2d mapping," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 22–29.

[12] M. Lourakis and A. Argyros, "Is levenberg-marquardt the most efficient optimization algorithm for implementing bundle adjustment?" in *IEEE International Conference on Computer Vision (ICCV)*, vol. 2. IEEE, 2005, pp. 1526–1531.

[13] I. Mahon, S. Williams, O. Pizarro, and M. Johnson-Roberson, "Efficient view-based slam using visual loop closures," *Robotics, IEEE Transactions on*, vol. 24, no. 5, pp. 1002–1014, 2008.

[14] E. Moore, "The shortest path through a maze," in *Proceedings of the International Symposium on the Theory of Switching*, 1957, pp. 285–292.

[15] J. Neira and J. Tardós, "Data association in stochastic mapping using the joint compatibility test," *Robotics and Automation, IEEE Transactions on*, vol. 17, no. 6, pp. 890–897, 2001.

[16] G. Sibley, "Relative bundle adjustment," Department of Engineering Science, Oxford University, Tech. Rep, Tech. Rep., 2009.

[17] G. Sibley, C. Mei, I. Reid, and P. Newman, "Adaptive relative bundle adjustment," in *Robotics Science and Systems Conference*, 2009, pp. 1–8.

[18] P. Spira and A. Pan, "On finding and updating spanning trees and shortest paths," *SIAM Journal on Computing*, vol. 4, p. 375, 1975.

[19] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon, "Bundle adjustmenta modern synthesis," *Vision algorithms: theory and practice*, pp. 153–177, 2000.