Graph SLAM Sparsification With Populated Topologies Using Factor Descent Optimization

Joan Vallvé, Joan Solà, and Juan Andrade-Cetto

***Abstract*—Current solutions to the simultaneous localization and mapping (SLAM) problem approach it as the optimization of a graph of geometric constraints. Scalability is achieved by reducing the size of the graph, usually in two phases. First, some selected nodes in the graph are marginalized and then, the dense and nonrelinearizable result is sparsified. The sparsified network has a new set of relinearizable factors and is an approximation to the original dense one. Sparsification is typically approached as a Kullback–Liebler divergence (KLD) minimization between the dense marginalization result and the new set of factors. For a simple topology of the new factors, such as a tree, there is a closed form optimal solution. However, more populated topologies can achieve a much better approximation because more information can be encoded, although in that case iterative optimization is needed to solve the KLD minimization. Iterative optimization methods proposed by the state-of-art sparsification require parameter tuning that strongly affects their convergence. In this letter, we propose factor descent and noncyclic factor descent, two simple algorithms for SLAM sparsification that match the state-of-art methods without any parameters to be tuned. The proposed methods are compared against the state of the art with regard to accuracy and CPU time, in both synthetic and real world datasets.**

***Index Terms*—SLAM, mapping, localization.**

I. Introduction

SIMULTANEOUS localization and mapping (SLAM) is the problem of building a map of the environment whilst localizing in it. One of its biggest pitfalls is that the problem grows over time: SLAM suffers from scalability. To tackle such growing resource demands, efforts have been invested mainly in two directions: by improving the efficiency of the algorithms, and by reducing the problem size. Despite recent improvements in algorithm efficiency [1], [2], the later is still of concern, as the complexity of the solution is always linked to the size of the problem. Therefore, methods for reducing the problem size while keeping as much information as possible are essential, especially for large SLAM experiments.

Manuscript received September 10, 2017; accepted January 3, 2018. Date of publication January 25, 2018; date of current version February 8, 2018. This letter was recommended for publication by Associate Editor M. Walter and Editor C. Stachniss upon evaluation of the reviewers' comments. This work was supported in part by the Spanish Ministry of Economy and Competitiveness under Projects ROBINSTRUCT (TIN2014-58178-R) and EB-SLAM (DPI2017-89564-P), in part by the EU H2020 Project LOGIMATIC (H2020-Galileo-2015-1-687534), and in part by the Spanish State Research Agency through the María de Maeztu Seal of Excellence to IRI MDM-2016-0656. *(Corresponding author: Joan Vallvé.)*

The authors are with the Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Barcelona 08028, Spain (e-mail: jvallve@iri.upc.edu; jsola@iri.upc.edu; cetto@iri.upc.edu).

Digital Object Identifier 10.1109/LRA.2018.2798283

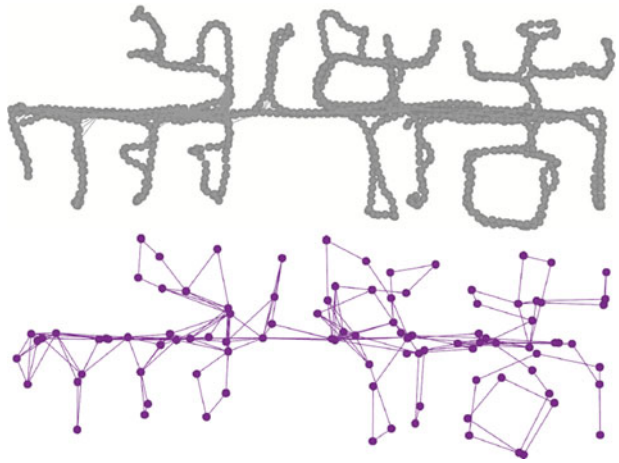

Fig. 1. Original (top) and sparsified graph (bottom) with 90% of node reduction in the Freiburg building dataset.

Several SLAM methods include mechanisms to limit the problem size growth. One of the simplest approaches consists in uniformly limiting the number of poses with respect to time or distance traveled, or to marginalize new poses close to old ones, thus growing only with the size of the area being mapped [3]. Carlone and Karaman [4] process only the most informative features in visual-inertial navigation by anticipating their visibility and future contribution to the estimation.

Information metrics can also be used to limit the size of the problem. For instance, Pose SLAM [5] only keeps new observations and robot poses if their entropy-based information content is significant. Vial *et al.* [6] proposed a conservative sparsification based on Kullback-Liebler divergence (KLD) for a filter-based SLAM. The sparsification is directly performed over the information matrix instead of creating a set of new factors. Kretzschmar and Stachniss [7] present an information-based compression method for laser-based pose graph SLAM, in which they compute a subset of nodes containing the scans that maximize the mutual information of the map for that subset. Chouldhary *et al.* [8] also propose to discard some landmarks depending on their information content using an entropy-based cost function.

Khosoussi *et al.* face the issue from a different perspective resorting to graph theory [9]. Under some assumptions, the weighted number of spanning trees of a graph approximates the determinant of the state covariance and can be used for measurement selection and pruning.

Most of the efficient SLAM methods take profit of two important characteristics of SLAM: sparsity and the capability of relinearization. Sparsity arises naturally from the fact that sensor

readings establish only a local subset of geometric constraints between variables, leaving most variables unconnected. Maintaining sparsity is important so that a computationally efficient solution to the problem can be found. On the other hand, since SLAM is a non-linear problem that is typically linearized to solve it, methods with the capability of relinearization greatly improve the accuracy of the solution.

In general, node marginalization is the only way of reducing the problem size without loss of information. However, marginalization has the disadvantage of causing loss of sparsity, increasing computational cost, and does not allow for relinearization, damping the accuracy.

In the graph-SLAM context, sparsification is the process of finding the best sparse and relinearizable approximation to the result of marginalization. The impact of sparsification can be observed in Fig. 1, where we show the result of marginalization and sparsification of the Freiburg building dataset: the sparsified problem (b) has only 10% of the nodes of the original problem (a), yet it captures almost the same information, yielding a very accurate solution.

Different approaches pose sparsification as a KLD minimization problem [10]–[12]. The best sparse approximation is the one with a minimum KLD with respect to the dense distribution resulting from marginalization. In case of using the simplest topology, i.e., a spanning tree, there exists a closed form for the optimal solution of all factors. However, in the majority of cases a tree topology is too simple to accurately approximate the dense result of node marginalization [13]. For richer (more populated) topologies, an iterative optimization is needed to solve the KLD minimization. This is the focus of the present paper.

State of the art methods [10]–[12] propose the use of interior point and projected quasi-Newton optimization methods to achieve sparsification. Both methods require the tuning of a set of parameters that strongly affects their convergence and robustness.

We presented in [13] a new optimization method for sparsification, named factor descent (FD). FD iteratively optimizes each of the factors leaving the rest of them fixed. For each factor, we compute the mean and information matrix that minimize the KLD given the rest of topology factors. The main advantage of factor descent is that it is a simple algorithm that does not require any tuning.

In this letter, we formalize and extend the formulation of our factor descent in [13], and provide analytic proofs for all the derivations. We also present a novel non-cyclic factor descent variant which exhibits faster convergence. Furthermore, in Section IV we explore a new periodic multi-node scheme for simultaneously removing sets of connected nodes that is more efficient in sparse SLAM problems. Our new C++ implementation validates and even enhances our preliminary results obtained by our Matlab prototype [13], as reported in the results Section V.

II. Node Removal and Sparsification in Graph SLAM

In graph-based SLAM, the problem is represented as a graph where the nodes refer to the variables and the factors (or edges) represent the geometrical constraints between variables. The state $\mathbf{x}$ is modeled as a multi-variate Gaussian distribution, and can include poses of the vehicle along its trajectory, some map representation or any sensor parameter. For each factor, we can define an error $\mathbf{e}$ as the discrepancy between a measurement $\mathbf{z}$ and its expectation,

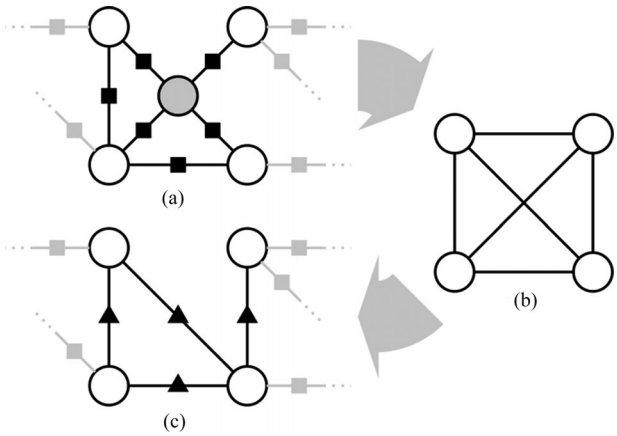

Fig. 2. Example of marginalization and sparsification of a node (gray). Triangle factors are the sparsification result.

$$\mathbf{e}(\mathbf{x}) = h(\mathbf{x}) - \mathbf{z} + \mathbf{v}, \qquad \mathbf{v} \sim \mathcal{N}\left(0, \mathbf{\Omega}^{-1}\right) \tag{1}$$

being $h(\mathbf{x})$ the sensor measurement model and $\mathbf{\Omega}$ the information matrix of the measurement Gaussian noise $\mathbf{v}$.

The maximum a posteriori estimation is obtained by iteratively minimizing the Mahalanobis squared norm of all linearized errors

$$\Delta\mathbf{x}^* = \arg\min_{\Delta\mathbf{x}} \sum_k \left\| h_k(\mathbf{x}) - \mathbf{z}_k + \mathbf{J}_k \Delta\mathbf{x} \right\|^2_{\mathbf{\Omega}_k^{-1}} \tag{2}$$

being $\mathbf{x}$ the state estimate at the current iteration, and $\mathbf{J}_k$ the Jacobian of the k-th measurement[1]. Until convergence, the optimal step $\Delta\mathbf{x}^*$ is used to update the expectations $h_k(\mathbf{x})$ and the Jacobians $\mathbf{J}_k$ and (2) is solved again. Current methods use Cholesky [2], [15] or QR [1], [16] matrix factorizations to solve for $\Delta\mathbf{x}^*$. Incremental methods [1], [2], update the problem directly on the factorized matrix obtaining important speed-ups. However, linearization errors are accumulated and the problem should be often rebuilt partially or completely.

Usually, reducing the SLAM problem size is approached in two different steps: node marginalization and sparsification (see Fig. 2). These two processes can be decoupled, postponing the second one depending on the available computational resources [11].

The whole process is faced locally. Once a node is selected for removal, the local problem is constrained over the immediate surroundings to that node, i.e., the node's Markov blanket (all nodes at distance 1) and all its intra-factors (the factors involving only nodes in the Markov blanket) as shown in Fig. 2(a). Optionally, this cropped problem can be solved using (2). Then, the new solution can be used henceforth, yielding slightly better results especially in on-line cases [12]. The selected node is marginalized via Schur complement, generating a dense information matrix $\mathbf{\Lambda}$, as in Fig. 2(b).

The aim of the sparsification process is to approximate the dense distribution $p(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma} = \mathbf{\Lambda}^{-1})$, resulting from node marginalization, with a sparse distribution $q(\mathbf{x}) \sim$

[1]In case of manifolds, (1) and the squared Mahalanobis norm in (2) become $\mathbf{e}(\mathbf{x}) = h(\mathbf{x}) \ominus \mathbf{z} \oplus \mathbf{v}$ and $\|h_k(\mathbf{x}) \ominus \mathbf{z}_k + \mathbf{J}_k \Delta\mathbf{x}\|^2_{\mathbf{\Omega}_k^{-1}}$, respectively, with $\mathbf{J}_k = \partial(h_k(\mathbf{x}) \ominus \mathbf{z}_k)/\partial\Delta\mathbf{x}$. The $\oplus$ and $\ominus$ are the addition and subtraction operators on the manifold, as described in [14].

$\mathcal{N}(\breve{\boldsymbol{\mu}}, \breve{\boldsymbol{\Sigma}})$ defined by a new set of factors as in Fig. 2(c). Sparsification is also split in two phases: building a topology, i.e., defining a set of new factors with their measurement model; and factor recovery, i.e., computing their mean and information.

A. Topology

As most of SLAM graphs, the topology is usually made up of factors with relative measurements between pairs of nodes. The simplest topology using relative measurements is a spanning tree. The tree that encodes the most information from the dense distribution is the Chow-Liu Tree (CLT). It is obtained from the tree with factors between the nodes with most mutual information. However, a tree topology is generally too sparse to approximate the original distribution.

More populated topologies can be built departing from the CLT and adding more factors, also based on the mutual information between nodes. This is sometimes called a sub-graph (SG) topology, as in [12]. Alternatively, departing from the CLT again, the cliquey topology [12] converts pairs of independent factors into one single factor by correlating them. In order to gather density, while SG adds more sparse factors, a cliquey topology densifies the existing ones.

Apart from CLT-based methods, a topology can be computed using a ℓ_1-regularized KLD minimization [11].

B. Factor Recovery Through KLD Minimization

Factor recovery computes the means $\breve{\mathbf{z}}_k$ and information $\breve{\boldsymbol{\Omega}}_k$ of all new factors of a given topology. We want those values that minimize the KLD between the dense $p(\mathbf{x})$ and sparse $q(\mathbf{x})$ distributions,

$$D_{KL} = \frac{1}{2}\left(\left\langle \breve{\boldsymbol{\Lambda}}, \boldsymbol{\Sigma}\right\rangle - \ln\left|\breve{\boldsymbol{\Lambda}}\boldsymbol{\Sigma}\right| + \left\|\breve{\boldsymbol{\mu}} - \boldsymbol{\mu}\right\|^2_{\breve{\boldsymbol{\Lambda}}^{-1}} - d\right), \quad (3)$$

where $\langle\cdot,\cdot\rangle$ denotes the matrix inner product.

Setting the expected measurement $\breve{\mathbf{z}}_k$ considering the dense distribution mean $\breve{\mathbf{z}}_k = h_k(\boldsymbol{\mu})$, the Mahalanobis norm term $\|\breve{\boldsymbol{\mu}} - \boldsymbol{\mu}\|^2_{\breve{\boldsymbol{\Lambda}}}$ becomes null. The rest of the expression can be simplified as follows. The dimension d of both distributions is constant. The log term is decomposed as $\ln|\breve{\boldsymbol{\Lambda}}\boldsymbol{\Sigma}| = \ln|\breve{\boldsymbol{\Lambda}}| + \ln|\boldsymbol{\Sigma}|$, whose second term is also constant w.r.t. all measurements information matrices $\breve{\boldsymbol{\Omega}}_k$. The information matrix of the approximate distribution can be expressed as $\breve{\boldsymbol{\Lambda}} = \breve{\mathbf{J}}^\top\breve{\boldsymbol{\Omega}}\breve{\mathbf{J}} = \sum_i \breve{\mathbf{J}}_i^\top\breve{\boldsymbol{\Omega}}_i\breve{\mathbf{J}}_i$, being $\breve{\boldsymbol{\Omega}}$ the block diagonal matrix containing all new factors' information matrices, $\breve{\boldsymbol{\Omega}} = \mathrm{diag}(\breve{\boldsymbol{\Omega}}_1 \dots \breve{\boldsymbol{\Omega}}_k \dots)$, and $\breve{\mathbf{J}} = [\breve{\mathbf{J}}_1^\top \dots \breve{\mathbf{J}}_k^\top \dots]^\top$ all new factors' Jacobians stacked. With all these considerations, the factors' information that minimizes (3) can be written as the constrained problem

$$\breve{\boldsymbol{\Omega}}^* = \arg\min_{\breve{\boldsymbol{\Omega}}} \left\langle \breve{\mathbf{J}}^\top\breve{\boldsymbol{\Omega}}\breve{\mathbf{J}}, \boldsymbol{\Sigma}\right\rangle - \ln\left|\breve{\mathbf{J}}^\top\breve{\boldsymbol{\Omega}}\breve{\mathbf{J}}\right|$$

$$\text{s.t. } \breve{\boldsymbol{\Omega}} = \mathrm{diag}\big(\breve{\boldsymbol{\Omega}}_1 \dots \breve{\boldsymbol{\Omega}}_k \dots\big), \breve{\boldsymbol{\Omega}} \succ 0. \quad (4)$$

In some cases such as original problems containing only relative measurements, the dense problem has a rank-deficient information matrix $\boldsymbol{\Lambda}$, so the covariance matrix $\boldsymbol{\Sigma}$ is not defined. In such case, a projection $\boldsymbol{\Lambda} = \mathbf{U}\mathbf{D}\mathbf{U}^\top$ such that $\mathbf{D}$ is invertible can be applied. Then, the KLD minimization in (4) in the reduced space can be performed by substituting

$$\breve{\mathbf{J}} \mapsto \breve{\mathbf{J}}\mathbf{U}, \quad \boldsymbol{\Sigma} \mapsto \mathbf{D}^{-1}. \quad (5)$$

To obtain the projection, one can re-parametrize the problem to relative poses w.r.t an arbitrarily chosen node [10], [11] or use a rank-revealing eigen decomposition [12].

C. Factor Recovery in Closed Form

According to [12], when the stacked Jacobian $\breve{\mathbf{J}}$ is invertible, the solution to (4) is obtained by imposing a null derivative w.r.t. all factor information matrices,

$$\breve{\boldsymbol{\Omega}}_k = \left(\breve{\mathbf{J}}_k\boldsymbol{\Sigma}\breve{\mathbf{J}}_k^\top\right)^{-1}. \quad (6)$$

This is the case, for instance, of the CLT topology in SLAM of relative measurements after applying the projection (5). However, and as has been said, the CLT topology can be too sparse to accurately approximate the exact dense distribution. The closed form (6) can also be used for the case of the cliquey topology. However, the cliquey breaks the homogeneity of measurement models, which is valuable in many cases.

D. Factor Recovery via Iterative Optimization

Other more populated topologies do not admit a closed form solution for all factors, and (4) has to be solved using iterative optimization. The state-of-the-art literature [10]–[12] proposes two different optimization methods for the factor recovery problem: Interior Point (IP) and Limited-memory Projected Quasi-Newton (PQN) [17].

In IP, the constraint of positive definiteness of the solution is included in the cost function as a log barrier,

$$\left\langle \breve{\mathbf{J}}^\top\breve{\boldsymbol{\Omega}}\breve{\mathbf{J}}, \boldsymbol{\Sigma}\right\rangle - \ln\left|\breve{\mathbf{J}}^\top\breve{\boldsymbol{\Omega}}\breve{\mathbf{J}}\right| - \rho\ln\left|\breve{\boldsymbol{\Omega}}\right|. \quad (7)$$

A stricter constraint can be applied instead of the log barrier term to also guarantee conservativeness: $\rho\ln|\boldsymbol{\Lambda} - \breve{\mathbf{J}}^\top\breve{\boldsymbol{\Omega}}\breve{\mathbf{J}}|$ [11]. The IP method consists of two nested loops. For each value of the log barrier parameter ρ_i, the resulting problem (7) is solved in the inner loop. After inner loop convergence, the outer loop decreases the log barrier parameter, $\rho_{i+1} = \alpha\rho_i$. The outer loop ends when ρ_i is 'close enough' to 0. The inner loop can be solved with Newton's method using the gradient and Hessian of (7) provided in [12].

The IP tuning parameters α and ρ_0, together with the inner loop's end conditions, strongly affect its convergence and robustness. To ensure that the positive definite constraint is satisfied, the contributions to the gradient and the Hessian of the KLD and the log barrier terms have to be balanced. Relaxing the inner loop end conditions or enhancing the decrease factor α lead to a lower contribution of the log barrier. This speeds up the method but may converge to a non positive definite result. In other words, tuning the IP parameters is a trade off between convergence velocity and robustness to divergence. This tuning is oftentimes problem-dependent.

In PQN, the positive definiteness constraint is accomplished through the projection along the line search onto the positive semi-definite subspace, by setting all negative eigenvalues to zero. PQN does not require the computation and inversion of the Hessian, but it evaluates the cost function in (4) several times at each iteration. Since the Markov Blanket is of small size, the evaluation of (4) is as costly as the Hessian computation, and IP greatly outperforms PQN both in time and convergence. Our preliminary results already showed it in terms of optimization iterations [13].

Algorithm 1: Factor descent sparsification.

Input: Dense mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$, topology $\mathcal{Z}$
Output: All factors' mean $\breve{\mathbf{z}}_k$ and information $\breve{\boldsymbol{\Omega}}_k$
// Precompute constant variables
for $\mathbf{z}_k \in \mathcal{Z}$ **do**
  $\mathbf{J}_k \leftarrow evaluateJacobian(\boldsymbol{\mu})$
  $\boldsymbol{\Phi}_k \leftarrow (\breve{\mathbf{J}}_k \boldsymbol{\Sigma} \breve{\mathbf{J}}_k^\top)^{-1}$
  $\breve{\mathbf{z}}_k \leftarrow h_k(\boldsymbol{\mu})$
end for
while not $endConditions()$ **do**
  // k-th factor descent
  $k \leftarrow nextFactor()$
  $\breve{\boldsymbol{\Upsilon}}_k \leftarrow \sum_{i \neq k} \breve{\mathbf{J}}_i^\top \breve{\boldsymbol{\Omega}}_i \breve{\mathbf{J}}_i$
  if $\exists \breve{\boldsymbol{\Upsilon}}_k^{-1}$ **then**
    $\breve{\boldsymbol{\Omega}}_k \leftarrow \boldsymbol{\Phi}_k - (\breve{\mathbf{J}}_k \breve{\boldsymbol{\Upsilon}}_k^{-1} \breve{\mathbf{J}}_k^\top)^{-1}$
  else
    $\breve{\boldsymbol{\Omega}}_k \leftarrow \boldsymbol{\Phi}_k - L^{-\top}\mathbf{Q}_L\big(\breve{\boldsymbol{\Upsilon}}_k - \breve{\boldsymbol{\Upsilon}}_k \mathbf{Q}_0^\top (\mathbf{Q}_0 \breve{\boldsymbol{\Upsilon}}_k \mathbf{Q}_0^\top)^{-1} \mathbf{Q}_0 \breve{\boldsymbol{\Upsilon}}_k\big)\mathbf{Q}_L^\top L^{-1}$
  end if
  // Ensure positive definite solution
  if $\breve{\boldsymbol{\Omega}}_k \prec 0$ **then**
    $\mathbf{V}, \boldsymbol{\lambda} \leftarrow eigenDecomposition(\breve{\boldsymbol{\Omega}}_k)$
    $\breve{\boldsymbol{\Omega}}_k \leftarrow \mathbf{V} diag(max(\epsilon, \boldsymbol{\lambda}))\mathbf{V}^\top$ // ϵ is a small positive
  end if
end while

An initial guess for relative measurements was proposed in [18] based on the off-diagonal blocks of the dense information matrix. Such initial guess is better than the identity matrix often used by default. Moreover, and contrary to what is stated in [12], it can be used for IP with the appropriate initial ρ.

The gradient of (7) can be split in two terms, the KLD term and the positive definite constraint term, $\nabla = \nabla_{KLD} + \rho \nabla_{PD}$. To balance their respective contributions we take a weighted ratio between both terms' norms, obtaining a warm start $\rho_0 = \omega \|\nabla_{KLD}\| / \|\nabla_{PD}\|$.

III. Factor Recovery With Factor Descent

Our proposed method Factor Descent (FD) is a sparsification optimization approach for solving (4) inspired in the coordinate descent optimization method: FD is a block-coordinate descent method. Each step consists in solving for a block of variables (those defining one factor's information matrix $\breve{\boldsymbol{\Omega}}_k$) while fixing the rest. So, at each step, (4) becomes

$$\breve{\boldsymbol{\Omega}}_k^* = \underset{\breve{\boldsymbol{\Omega}}_k}{\arg\min} \left\langle \breve{\boldsymbol{\Upsilon}}_k + \breve{\mathbf{J}}_k^\top \breve{\boldsymbol{\Omega}}_k \breve{\mathbf{J}}_k, \boldsymbol{\Sigma} \right\rangle - \ln \left| \breve{\boldsymbol{\Upsilon}}_k + \breve{\mathbf{J}}_k^\top \breve{\boldsymbol{\Omega}}_k \breve{\mathbf{J}}_k \right| \quad \text{s.t.} \quad \breve{\boldsymbol{\Omega}}_k \succ 0 \tag{8}$$

where $\breve{\boldsymbol{\Upsilon}}_k$ is the information matrix considering only the rest of the factors in the topology,

$$\breve{\boldsymbol{\Upsilon}}_k = \sum_{i \neq k} \breve{\mathbf{J}}_i^\top \breve{\boldsymbol{\Omega}}_i \breve{\mathbf{J}}_i. \tag{9}$$

Descent of the KLD cost is achieved factor by factor, and hence the Factor Descent name. When all factors other than k are unchanged, the optimal $\breve{\boldsymbol{\Omega}}_k$ can be computed analytically by finding the null derivative of (8). This can be done in different manners depending on the particular properties of $\breve{\boldsymbol{\Upsilon}}_k$ and $\breve{\mathbf{J}}_k$. Consider the following propositions (see proofs in the Appendix).

Proposition 1: If $\breve{\boldsymbol{\Lambda}}$ is invertible and $\breve{\mathbf{J}}_k$ is full rank, the derivative of (8) is null in

$$\breve{\boldsymbol{\Omega}}_k = \left(\breve{\mathbf{J}}_k \boldsymbol{\Sigma} \breve{\mathbf{J}}_k^\top\right)^{-1} - L^{-\top}\mathbf{Q}_L\big(\breve{\boldsymbol{\Upsilon}}_k - \breve{\boldsymbol{\Upsilon}}_k \mathbf{Q}_0^\top \left(\mathbf{Q}_0 \breve{\boldsymbol{\Upsilon}}_k \mathbf{Q}_0^\top\right)^{-1} \mathbf{Q}_0 \breve{\boldsymbol{\Upsilon}}_k\big)\mathbf{Q}_L^\top L^{-1} \tag{10}$$

being the LQ-decomposition of $\breve{\mathbf{J}}_k = \mathbf{L}\mathbf{Q} = [L \;\; 0] \begin{bmatrix} \mathbf{Q}_L \\ \mathbf{Q}_0 \end{bmatrix}$.

Proposition 2: If $\breve{\boldsymbol{\Upsilon}}_k$ is invertible and $\breve{\mathbf{J}}_k$ is full rank, the derivative of (8) is null in

$$\breve{\boldsymbol{\Omega}}_k = \left(\breve{\mathbf{J}}_k \boldsymbol{\Sigma} \breve{\mathbf{J}}_k^\top\right)^{-1} - \left(\breve{\mathbf{J}}_k \breve{\boldsymbol{\Upsilon}}_k^{-1} \breve{\mathbf{J}}_k^\top\right)^{-1}. \tag{11}$$

Furthermore, (11) can be computed efficiently by using the Cholesky decomposition of $\breve{\boldsymbol{\Upsilon}} = \mathbf{R}^\top \mathbf{R}$

$$\breve{\boldsymbol{\Omega}}_k = \left(\breve{\mathbf{J}}_k \boldsymbol{\Sigma} \breve{\mathbf{J}}_k^\top\right)^{-1} - \left(\boldsymbol{\Gamma}\boldsymbol{\Gamma}^\top\right)^{-1}, \tag{12}$$

where $\boldsymbol{\Gamma} = \breve{\mathbf{J}}_k \mathbf{R}^{-1}$ is directly obtained by back substitution.

Prop. 1 applies to all cases if one takes care to project the subgraph with (5), but its computation is expensive. It complements and generalizes our previous formulation [13]. Prop. 2 applies to the cases where the subgraph with the current k factor removed would still be full rank. Cases where Prop. 2 does not apply, and therefore we must resort to Prop. 1, include e.g., landmarks observed from two monocular views (removing one view's factor renders the landmark's depth unobservable) or constrained IMU motion factors (removing the constraining factors renders the IMU biases unobservable).

The method is described in Alg. 1. Since the first term of both (10) and (12) does not depend on the rest of factors, it can be computed only once. This term can be interpreted as the projection, onto the k-th factor's measurement space, of the information of the dense distribution resulting of the node marginalization. Analogously, the second term in both cases is the projection, onto the measurement space of the k-th factor, of the information of the rest of the factors.

The projection (5) can be applied in case of a rank-deficient $\boldsymbol{\Lambda}$ as well (note that the rest of new factor's information matrix $\breve{\boldsymbol{\Upsilon}}_k$ must be projected too). In this case, assuming $\breve{\boldsymbol{\Lambda}}$ is invertible is equivalent to assuming that the rank has not decreased, $rk(\breve{\boldsymbol{\Lambda}}) = rk(\boldsymbol{\Lambda})$. Also, a full rank Jacobian $\breve{\mathbf{J}}_k$ only implies linear independence of all measurement elements. With these considerations, we can ensure that all assumptions are taken without loss of generality.

A. Closed Form Factors

In some cases, the second term of (10) is null, yielding the same closed form solution (6) presented in Section II-C.

Proposition 3: If $\breve{\boldsymbol{\Lambda}}$ is invertible, $\breve{\mathbf{J}}_k$ is full rank and $\text{nul}(\breve{\boldsymbol{\Upsilon}}_k) = \text{rank}(\breve{\mathbf{J}}_k)$, the derivative of (8) is null in (6).

Prop. 3 applies to those factors whose Jacobians $\breve{\mathbf{J}}_k$ are linearly independent of all the rest of factors' Jacobians of the topology. The closed form (6) introduced in [12] is then the optimal solution. This happens, for instance, for a factor such that

without it the topology becomes disconnected. Trivially, this is the case for tree topologies as described in [12]. For general topologies, (6) is applicable to those factors that fulfill the condition, and the optimization is only needed to solve for the rest. Therefore, we want to emphasize that (10) is a generalization of the closed form solution in [12].

The presented formulation amends our prior work [13] where we applied Prop. 3 in case of not invertible $\breve{\mathbf{\Upsilon}}_k$. This is only true in pose-graph SLAM with factors with strictly positive definite information.

B. Positive-Definiteness

The solutions (10) and (12) are based on the KLD derivatives, and no positive definiteness constraint is applied. Then, the result can be a non positive definite solution if the second term of (10) or (12) is larger than the first term, that is, if the projection of the information of the rest of factors has a larger information content than the projection of the dense distribution. In other words, when the approximation made by the rest of factors $\breve{\mathbf{\Upsilon}}_k$ is not conservative in some direction, the optimal k-factor would subtract this excess of information, leading to a negative eigenvalue of $\breve{\mathbf{\Omega}}_k$. In this case, we set all negative eigenvalues to a small positive value.

C. Non-Cyclic Factor Descent

Factor Descent iterates over all factors cyclically. Clearly, the order in which the factors are optimized can be altered to our benefit. To improve convergence, we propose selecting at each step the factor that will decrease the KLD the most. To find it we compute the gradient of the KLD w.r.t. each non-zero element of $\breve{\mathbf{\Omega}}$. Each factor's information $\breve{\mathbf{\Omega}}_k$ has its corresponding gradient segment. We select the one with the largest norm as the one that would reduce the KLD the most.

IV. Multi-Node Marginalization and Sparsification

Typically, the marginalization and sparsification is done sequentially, node by node. The sequential scheme is the only possible alternative for online marginalization and sparsification. However, if the marginalization and sparsification of the selected nodes is made periodically, different alternative schemes appear.

Multi-node marginalization and sparsification would also be possible, considering groups of nodes at a time —for instance, those connected by one single factor. In the multi-node scheme, neither the procedure for marginalization nor sparsification suffer any changes. Taking as the Markov blanket the union of all removed nodes' Markov blankets, the marginalization of the group of nodes leads to a dense problem in the exact same way as if removing a single node.

When applicable, this is better than proceeding sequentially, since the sequential procedure generates accumulation of approximations given that some factors resulting from a sparsification become intra-factors of the next node to be removed. Fig. 3 depicts a toy example of the two schemes for the removal of two nodes connected by a factor. Note how in the sequential scheme, three factors (marked with a grey area) resulting from the first sparsification become intra-factors in the second one.

Normally, connected nodes share most of their Markov blankets, and the multi-node Markov blankets are only slightly larger than the individual ones, depending on the sparsity of the graph. This derives in faster computation of the multinode scheme, since the process is executed once for the entire group of nodes instead of sequentially one by one.

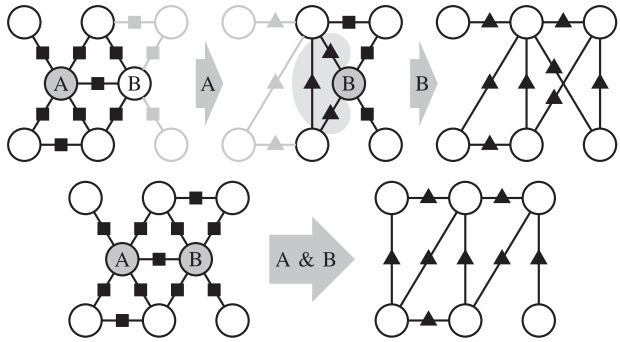

Fig. 3. Example of sequential (top) and multi-node (bottom) schemes for removing and sparsifiying nodes A and B.

The resulting topology in the multi-node scheme is usually different than that of the sequential method. While the first is designed considering all the connected nodes' Markov Blankets, the second one is an accumulation of locally designed topologies. For example, when sparsifying with tree topologies such as CLT, the resulting topology in multi-node is indeed a tree, while the accumulation of trees in the sequential scheme usually yields a denser topology. For this reason, the multi-node scheme not always produces more accurate approximations in terms of KLD.

V. Results

Our preliminary results in [13] were based on the number of optimization iterations instead of CPU computation time. This is because that implementation was unoptimized and in Matlab, and therefore CPU time measurements were not reliable. In order to rigorously test the performance of our proposed methods, we re-implemented factor descent and Interior Point in C++. The novel non-cyclic factor descent and the multi-node scheme were also implemented. We use our own non-linear least squares SLAM implementation based on Ceres [19]. Due to the bad convergence rates and computational costs obtained in [13] for PQN, it has been discarded as a suitable method for sparsification.

A. Convergence

In a first test, we evaluate the convergence in CPU time of the three iterative sparsification methods: factor descent (FD), non-cyclic factor descent (ncFD) and interior point (IP). To guarantee equal conditions for all the methods under test, we executed a SLAM for the Manhattan M3500 dataset [20] with 80% of node removal and stored all sparsification problems after each node marginalization. Afterwards, all stored problems were solved by the three methods to compare their convergence rates.

The parameters for the IP method are explained in Section II-D, and were set as a result of a delicate tuning process. Specifically, we set the ρ decrease parameter $\alpha = 0.5$. The balance weight $\omega = 0.1$ is used for setting the initial value ρ_0. Also, we imposed a relaxed end condition for the inner loop when the norm of the KLD gradient becomes lower than 1.

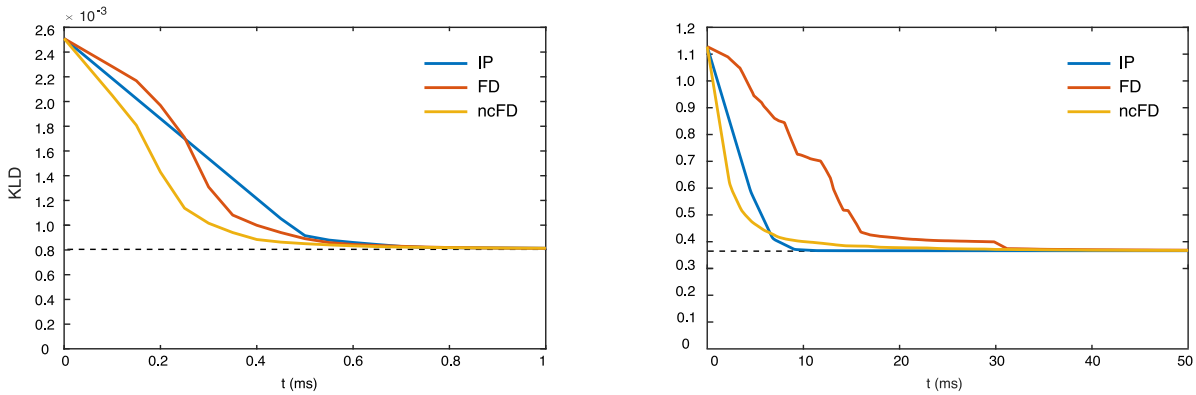

Fig. 4. Mean KLD evolution of the compared methods, for all problems of Markov blanket size 3 (left) and 8 (right) in the Manhattan experiment with 80% of node removal. Note the different KLD and time scales.

Conversely, neither factor descent nor non-cyclic factor descent have any internal parameter to be tuned.

Fig. 4 shows the mean KLD evolution of each sparsification method for all problems of Markov blanket size 3 and 8. The initialization based on the dense information off-diagonal blocks is closer to the optimal solution for small problems than for bigger ones. Likewise, all methods converge faster for small problems since IP has smaller Hessian and FD and ncFD have less factors to iterate over. The convergence of FD and ncFD are comparable to IP's. Additionally, the benefits of the non-cyclic strategy are clear, specially in bigger problems.

B. Application

Our second battery of tests is made with the purpose of evaluating the performance of each method and the multi-node scheme in a real application.

We tested each method using both sequential and the multi-node scheme on four different datasets [20] with different node reduction levels. Since node selection is out of the scope of this letter, we applied the simple strategy of keeping one node every N. The typology of the chosen datasets is very different. The Manhattan M3500 sequence is large and dense (i.e. highly connected), which means large Markov blankets. On the contrary, the Killian Court dataset has few loop closures leading to small Markov blankets. The Freiburg Building (FR079) and Intel Research Lab sequences are a compromise between the other two datasets. Much denser datasets such as the city10k constitute a challenge for our and the other methods, and are considered for future work.

We compare four sparsification methods. First, CLT with closed form sparsification. Second, an SG topology with twice as factors as CLT as described in Section II-A. Three iterative optimization methods are compared using SG: IP, FD and ncFD. For these, we apply the same end condition: when all elements of the KLD gradient become lower than 10^{-3}. Also a maximum time condition is set to 50 ms.

An independent experiment is ran for each method. Node marginalization and sparsification is performed every 100 nodes. Then, each experiment accumulates the sparsification approximations along the whole dataset. The original SLAM graph without removing any node is taken as a baseline. The global KLD between each method and the baseline is computed using (3), but this time evaluating for the whole SLAM problem. As in [12], factors involving previously removed nodes were redirected to the closest existing node. In order not to distort the KLD results, this was also done for the baseline graph. The SLAM problem is relinearized continuously to prevent linearization errors to be confused with sparsification inaccuracy.

TABLE I
AMOUNT OF SPARSIFICATION PROBLEMS SOLVED (#) AND MARKOV BLANKET MEAN SIZE (n) IN SEQUENTIAL AND MULTI-NODE SCHEMES

Dataset	Density	Topology	Scheme	Node reduction							
				66.6%		75%		80%		90%	
				#	n	#	n	#	n	#	n
Manhattan	1.558	CLT	Seq.	2332	2.46	2624	2.42	2799	2.41	3149	2.33
			Multi	1177	2.91	909	3.18	734	3.45	384	4.21
		SG	Seq.	2332	2.71	2624	2.77	2799	2.90	3149	3.10
			Multi	1177	2.91	909	3.21	734	3.48	384	4.29
Intel	1.227	CLT	Seq.	818	2.23	920	2.25	981	2.26	1104	2.27
			Multi	413	2.41	319	2.57	258	2.75	135	3.19
		SG	Seq.	818	2.29	920	2.34	981	2.42	1104	2.64
			Multi	413	2.41	319	2.58	258	2.76	135	3.25
FR079	1.232	CLT	Seq.	658	2.22	741	2.20	790	2.20	889	2.20
			Multi	332	2.33	256	2.37	207	2.42	108	2.65
		SG	Seq.	658	2.26	741	2.27	790	2.29	889	2.36
			Multi	332	2.33	256	2.38	207	2.42	108	2.67
Killian	1.025	CLT	Seq.	538	2.02	605	2.04	645	2.04	726	2.07
			Multi	271	2.04	210	2.08	170	2.08	89	2.19
		SG	Seq.	538	2.02	605	2.05	645	2.06	726	2.12
			Multi	271	2.04	210	2.08	170	2.08	89	2.20

Table I contains the amount of sparsification problems solved and the Markov blanket mean size using sequential and multi-node schemes for all datasets-node reduction combinations. As can be observed, the multi-node scheme reduces significantly the amount of sparsifications performed —in exchange, there is an increase in the Markov blanket mean size, particularly for highly connected cases.

Table II includes the final global KLD values after applying each method in both sequential and multi-node schemes in the different datasets for different node reduction ratios.

All methods using SG topology achieve similar KLD and RMSE values in all experiments. However, the approximation of CLT is significantly worse. As stated before, the tree topology is too sparse to explain the dense distribution. The optimal parameters of IP are not the same for all datasets and node reduction levels, and a significant effort on tuning was needed to achieve the optimal KLD reduction with as less computational cost as possible. Conversely, the simplicity of the algorithm and the absence of parameters are the main advantages of FD

TABLE II
COMPARISON OF FINAL GLOBAL KLD AND CPU TIME FOR ALL METHODS, DIFFERENT DATASETS AND NODE REDUCTION LEVELS

Dataset	Scheme	Method	Node reduction											
			66.6%			75%			80%			90%		
			KLD	RMSE	Total time	KLD	RMSE	Total time	KLD	RMSE	Total time	KLD	RMSE	Total time
Manhattan	Seq.	CLT	59.39	0.297	0.23 s	45.79	0.607	0.26 s	32.33	0.113	0.28 s	17.32	0.258	0.28 s
		IP	3.72	0.037	8.49 s	2.58	0.026	11.41 s	2.93	0.036	14.77 s	3.17	0.138	25.51 s
		FD	**3.53**	0.044	3.30 s	**2.44**	**0.024**	4.91 s	**2.69**	0.048	7.02 s	**2.73**	0.148	16.25 s
		ncFD	3.56	0.046	2.38 s	2.46	0.024	2.97 s	2.69	0.035	3.90 s	2.75	0.136	13.52 s
	Multi	CLT	60.39	0.245	**0.17 s**	46.73	0.495	**0.16 s**	42.07	0.168	**0.16 s**	36.68	0.096	**0.13 s**
		IP	3.72	0.017	6.12 s	2.76	0.028	6.70 s	3.54	0.038	7.71 s	4.80	0.093	***13.00 s***
		FD	3.70	**0.015**	3.11 s	2.78	0.033	4.24 s	3.54	0.037	5.92 s	4.71	0.094	20.30 s
		ncFD	3.76	0.029	***2.15 s***	2.76	0.030	***2.29 s***	3.45	**0.030**	***3.46 s***	4.75	**0.091**	16.49 s
Intel	Seq.	CLT	28.77	0.094	0.07 s	29.00	0.048	0.08 s	29.16	0.066	0.09 s	17.47	0.118	0.09 s
		IP	6.04	0.022	1.93 s	5.66	0.044	2.21 s	5.42	0.022	2.85 s	2.51	0.022	4.69 s
		FD	5.45	0.024	1.68 s	6.79	0.049	1.81 s	**5.04**	0.023	2.20 s	2.04	0.023	2.55 s
		ncFD	5.46	0.024	1.33 s	5.87	0.046	1.35 s	5.15	0.024	1.62 s	2.14	0.023	1.59 s
	Multi	CLT	25.33	0.115	**0.04 s**	21.90	0.079	**0.04 s**	22.59	0.043	**0.04 s**	10.42	0.157	**0.02 s**
		IP	5.91	0.023	1.46 s	**4.78**	**0.028**	1.37 s	5.61	**0.015**	1.50 s	1.99	0.018	1.31 s
		FD	**5.27**	**0.021**	1.43 s	4.81	0.032	1.38 s	5.59	0.016	1.53 s	**1.93**	**0.018**	1.49 s
		ncFD	5.34	0.022	***1.11 s***	5.16	0.030	***1.00 s***	5.55	0.016	***0.96 s***	2.05	0.018	***0.92 s***
FR079	Seq.	CLT	12.93	0.024	0.05 s	13.73	0.015	0.06 s	12.92	0.028	0.07 s	10.11	0.025	0.07 s
		IP	**2.63**	0.008	1.91 s	2.27	0.005	2.15 s	1.66	0.003	2.71 s	1.16	0.009	3.75 s
		FD	2.64	**0.008**	1.81 s	2.33	0.004	2.05 s	**1.63**	0.004	2.54 s	0.98	0.010	3.29 s
		ncFD	2.68	0.008	1.52 s	2.31	0.005	1.59 s	1.71	**0.003**	2.11 s	0.99	**0.009**	2.63 s
	Multi	CLT	14.81	0.022	**0.03 s**	13.05	0.014	**0.03 s**	10.04	0.018	**0.02 s**	5.94	0.012	**0.01 s**
		IP	3.08	0.009	1.18 s	**2.14**	0.003	0.97 s	1.70	0.005	1.20 s	**0.97**	0.017	***0.92 s***
		FD	3.10	0.009	1.18 s	2.16	0.003	1.01 s	1.70	0.006	1.16 s	1.15	0.016	1.11 s
		ncFD	3.17	0.010	***0.92 s***	2.22	**0.003**	***0.78 s***	1.75	0.005	***0.95 s***	1.29	0.018	0.98 s
Killian	Seq.	CLT	2.48	0.520	0.03 s	6.43	0.290	0.04 s	7.92	1.048	0.04 s	9.51	2.887	0.05 s
		IP	0.37	0.230	0.10 s	0.43	**0.181**	0.22 s	2.18	0.580	0.26 s	0.41	0.473	1.24 s
		FD	0.37	0.230	0.05 s	0.45	0.192	0.07 s	2.19	0.571	0.08 s	0.41	0.459	0.26 s
		ncFD	0.37	0.229	0.04 s	0.42	0.182	0.07 s	2.18	0.569	0.08 s	0.41	0.385	0.26 s
	Multi	CLT	2.15	0.266	**0.02 s**	14.83	0.236	**0.01 s**	3.39	1.337	**0.01 s**	3.39	2.402	**0.01 s**
		IP	**0.08**	0.053	0.07 s	**0.36**	0.181	0.11 s	0.42	0.262	0.09 s	**0.28**	0.327	0.32 s
		FD	0.08	0.053	0.03 s	0.38	0.191	0.04 s	0.41	0.265	0.04 s	0.28	0.352	0.13 s
		ncFD	0.08	**0.053**	***0.03 s***	0.36	0.185	***0.03 s***	**0.41**	**0.255**	***0.03 s***	0.28	**0.304**	***0.10 s***

and ncFD. Furthermore, both FD and ncFD outperform IP in computational time in almost all the experiments.

A part from CLT, ncFD is faster than IP and FD in almost all experiments. As pointed out before, ncFD convergence improvements w.r.t. FD are specially relevant for the case of big Markov blankets. For this reason, the computation time benefits are more significant for the denser datasets. While in the Manhattan dataset the total time spent by ncFD is lower than the cyclic version, in the Killian dataset it is similar.

The multi-node scheme speeds up all methods by reducing the amount of sparsification problems to be solved. However, the Markov blanket growth may explain the different performance in KLD and RMSE depending on the dataset. In the Killian dataset, the multi-node scheme produces more accurate approximations than the sequential scheme for all methods and reduction levels. However, in the FR079 and Intel datasets, using multi-node instead of sequential is not beneficial in any of the cases regarding to KLD and RMSE. The Markov blanket growth in the Manhattan dataset is strong, undermining the approximation accuracy, especially for high node reduction levels.

VI. CONCLUSIONS AND FUTURE WORK

This letter presented the Factor Descent and non-cyclic Factor Descent optimization methods for the sparsification of populated topologies in large-scale graph-based SLAM. Our results show that both methods compete with the most popular state-of-art method (interior point) both in accuracy and computational time and even outperforms it in most cases. At the same time, the simplicity of the algorithm makes FD and its non-cyclic version ncFD an appealing approach compared with interior point method, which requires the tuning of parameters. We demonstrated convergence improvements of our novel non-cyclic version ncFD specially in highly connected problems. We also introduced the new multi-node scheme for periodic marginalization and sparsification that is more efficient especially in moderately connected problems.

In the course of our investigations we have encountered some convergence difficulties to treat much denser datasets such as the city10k. These have been observed with all the methods (FD, IP and PQN) and therefore this type of problems represents still an open challenge. We suspect the issues might be related to numerical stability, since the information content of all factors is very uneven, with differences of several orders of magnitude. We are currently investigating the causes for this poor performance in order to propose new solutions.

Also for future work we consider the application of our FD methods for SLAM sessions that include heterogeneous measurement sources such as image projection of 3D points or IMU measurements, including their biases.

APPENDIX

Proposition 1: If $\breve{\mathbf{\Lambda}}$ is invertible and $\breve{\mathbf{J}}_k$ is full rank, the derivative of (8) is null in

$$\breve{\mathbf{\Omega}}_k = \left(\breve{\mathbf{J}}_k \mathbf{\Sigma} \breve{\mathbf{J}}_k^\top\right)^{-1} - L^{-\top}\mathbf{Q}_L\big(\breve{\mathbf{\Upsilon}}_k - \breve{\mathbf{\Upsilon}}_k\mathbf{Q}_0^\top\left(\mathbf{Q}_0\breve{\mathbf{\Upsilon}}_k\mathbf{Q}_0^\top\right)^{-1}\mathbf{Q}_0\breve{\mathbf{\Upsilon}}_k\big)\mathbf{Q}_L^\top L^{-1} \tag{13}$$

being the LQ-decomposition of $\breve{\mathbf{J}}_k = \mathbf{L}\mathbf{Q} = [L\ \ 0]\begin{bmatrix}\mathbf{Q}_L\\ \mathbf{Q}_0\end{bmatrix}$.

Proof: The derivative of (8) w.r.t $\breve{\mathbf{\Omega}}_k$ is

$$\frac{\partial D_{KL}}{\partial \breve{\mathbf{\Omega}}_k} = \breve{\mathbf{J}}_k\mathbf{\Sigma}\breve{\mathbf{J}}_k^\top - \breve{\mathbf{J}}_k\left(\breve{\mathbf{\Upsilon}}_k + \breve{\mathbf{J}}_k^\top\breve{\mathbf{\Omega}}_k\breve{\mathbf{J}}_k\right)^{-1}\breve{\mathbf{J}}_k^\top. \tag{14}$$

Applying the decomposition into the second term:

$$\begin{aligned}
&\breve{\mathbf{J}}_k\left(\breve{\mathbf{\Upsilon}}_k + \breve{\mathbf{J}}_k^\top\breve{\mathbf{\Omega}}_k\breve{\mathbf{J}}_k\right)^{-1}\breve{\mathbf{J}}_k^\top\\
&= \mathbf{L}\mathbf{Q}\left(\breve{\mathbf{\Upsilon}}_k + \mathbf{Q}^\top\mathbf{L}^\top\breve{\mathbf{\Omega}}_k\mathbf{L}\mathbf{Q}\right)^{-1}\mathbf{Q}^\top\mathbf{L}^\top\\
&= \mathbf{L}\left(\mathbf{Q}\breve{\mathbf{\Upsilon}}_k\mathbf{Q}^\top + \mathbf{L}^\top\breve{\mathbf{\Omega}}_k\mathbf{L}\right)^{-1}\mathbf{L}^\top\\
&= [L\ \ 0]\begin{bmatrix} L^\top\breve{\mathbf{\Omega}}_k L + \mathbf{Q}_L\breve{\mathbf{\Upsilon}}_k\mathbf{Q}_L^\top & \mathbf{Q}_L\breve{\mathbf{\Upsilon}}_k\mathbf{Q}_0^\top\\ \mathbf{Q}_0\breve{\mathbf{\Upsilon}}_k\mathbf{Q}_L^\top & \mathbf{Q}_0\breve{\mathbf{\Upsilon}}_k\mathbf{Q}_0^\top\end{bmatrix}^{-1}\begin{bmatrix}L^\top\\ 0\end{bmatrix}\\
&= L\big(L^\top\breve{\mathbf{\Omega}}_k L + \mathbf{Q}_L\breve{\mathbf{\Upsilon}}_k\mathbf{Q}_L^\top\\
&\quad - \mathbf{Q}_L\breve{\mathbf{\Upsilon}}_k\mathbf{Q}_0^\top\left(\mathbf{Q}_0\breve{\mathbf{\Upsilon}}_k\mathbf{Q}_0^\top\right)^{-1}\mathbf{Q}_0\breve{\mathbf{\Upsilon}}_k\mathbf{Q}_L^\top\big)^{-1}L^\top\\
&= \big(\breve{\mathbf{\Omega}}_k + L^{-\top}\big(\mathbf{Q}_L\breve{\mathbf{\Upsilon}}_k\mathbf{Q}_L^\top\\
&\quad - \mathbf{Q}_L\breve{\mathbf{\Upsilon}}_k\mathbf{Q}_0^\top\left(\mathbf{Q}_0\breve{\mathbf{\Upsilon}}_k\mathbf{Q}_0^\top\right)^{-1}\mathbf{Q}_0\breve{\mathbf{\Upsilon}}_k\mathbf{Q}_L^\top\big)L^{-1}\big)^{-1}.
\end{aligned} \tag{15}$$

Substituting in (14) and imposing null derivative leads to (13). Since $\mathbf{Q}$ is orthogonal and $\mathbf{Q}_0\breve{\mathbf{\Lambda}}\mathbf{Q}_0^\top = \mathbf{Q}_0\breve{\mathbf{\Upsilon}}_k\mathbf{Q}_0^\top$, then $\exists\breve{\mathbf{\Lambda}}^{-1} \Rightarrow \exists(\mathbf{Q}_0\breve{\mathbf{\Lambda}}\mathbf{Q}_0^\top)^{-1} \Rightarrow \exists(\mathbf{Q}_0\breve{\mathbf{\Upsilon}}_k\mathbf{Q}_0^\top)^{-1}$.

Proposition 2: If $\breve{\mathbf{\Upsilon}}_k$ is invertible and $\breve{\mathbf{J}}_k$ is full rank, the derivative of (8) is null in

$$\breve{\mathbf{\Omega}}_k = \left(\breve{\mathbf{J}}_k\mathbf{\Sigma}\breve{\mathbf{J}}_k^\top\right)^{-1} - \left(\breve{\mathbf{J}}_k\breve{\mathbf{\Upsilon}}_k^{-1}\breve{\mathbf{J}}_k^\top\right)^{-1}. \tag{16}$$

Proof: If $\breve{\mathbf{\Upsilon}}_k$ is invertible, applying the Woodbury matrix identity forwards and backwards to the second term of (14)

$$\begin{aligned}
&\breve{\mathbf{J}}_k\left(\breve{\mathbf{\Upsilon}}_k + \breve{\mathbf{J}}_k^\top\breve{\mathbf{\Omega}}_k\breve{\mathbf{J}}_k\right)^{-1}\breve{\mathbf{J}}_k^\top\\
&= \breve{\mathbf{J}}_k\big(\breve{\mathbf{\Upsilon}}_k^{-1} - \breve{\mathbf{\Upsilon}}_k^{-1}\breve{\mathbf{J}}_k^\top\left(\breve{\mathbf{\Omega}}_k^{-1} + \breve{\mathbf{J}}_k\breve{\mathbf{\Upsilon}}_k^{-1}\breve{\mathbf{J}}_k^\top\right)^{-1}\breve{\mathbf{J}}_k\breve{\mathbf{\Upsilon}}_k^{-1}\big)\breve{\mathbf{J}}_k^\top\\
&= \breve{\mathbf{J}}_k\breve{\mathbf{\Upsilon}}_k^{-1}\breve{\mathbf{J}}_k^\top - \breve{\mathbf{J}}_k\breve{\mathbf{\Upsilon}}_k^{-1}\breve{\mathbf{J}}_k^\top\left(\breve{\mathbf{\Omega}}_k^{-1} + \breve{\mathbf{J}}_k\breve{\mathbf{\Upsilon}}_k^{-1}\breve{\mathbf{J}}_k^\top\right)^{-1}\breve{\mathbf{J}}_k\breve{\mathbf{\Upsilon}}_k^{-1}\breve{\mathbf{J}}_k^\top\\
&= \left(\left(\breve{\mathbf{J}}_k\breve{\mathbf{\Upsilon}}_k^{-1}\breve{\mathbf{J}}_k^\top\right)^{-1} + \breve{\mathbf{\Omega}}_k\right)^{-1}.
\end{aligned}$$

Substituting in (14) and imposing null derivative leads to (16).

Proposition 3: If $\breve{\mathbf{\Lambda}}$ is invertible, $\breve{\mathbf{J}}_k$ is full rank and $nul(\breve{\mathbf{\Upsilon}}_k) = rk(\breve{\mathbf{J}}_k)$, the derivative of (8) is null in (6).

Proof: Consider $\breve{\mathbf{J}}_k \in \mathbb{R}^{m\times n}, \breve{\mathbf{\Upsilon}} \in \mathbb{R}^{n\times n}, n > m$. Since $\breve{\mathbf{J}}_k$ is full rank, $rk(\breve{\mathbf{J}}_k) = m$. According to Prop. 1, $\exists(\mathbf{Q}_0\breve{\mathbf{\Upsilon}}_k\mathbf{Q}_0^\top)^{-1}$ and $rk(\mathbf{Q}_0\breve{\mathbf{\Upsilon}}_k\mathbf{Q}_0^\top) = n - m$. Since $\mathbf{Q}$ is orthogonal, $rk(\mathbf{Q}\breve{\mathbf{\Upsilon}}_k\mathbf{Q}^\top) = rk(\breve{\mathbf{\Upsilon}}_k) = n - nul(\breve{\mathbf{\Upsilon}}_k) = n - rk(\breve{\mathbf{J}}_k) = n - m$. According to the Schur complement rank additivity formula

$$\begin{aligned}
rk\left(\mathbf{Q}\breve{\mathbf{\Upsilon}}_k\mathbf{Q}^\top\right) &= rk\left(\mathbf{Q}_0\breve{\mathbf{\Upsilon}}_k\mathbf{Q}_0^\top\right) + rk\big(\mathbf{Q}_L\breve{\mathbf{\Upsilon}}_k\mathbf{Q}_L^\top\\
&\quad - \mathbf{Q}_L\breve{\mathbf{\Upsilon}}_k\mathbf{Q}_0^\top\left(\mathbf{Q}_0\breve{\mathbf{\Upsilon}}_k\mathbf{Q}_0^\top\right)^{-1}\mathbf{Q}_0\breve{\mathbf{\Upsilon}}_k\mathbf{Q}_L^\top\big),
\end{aligned}$$

then $\mathbf{Q}_L\breve{\mathbf{\Upsilon}}_k\mathbf{Q}_L^\top - \mathbf{Q}_L\breve{\mathbf{\Upsilon}}_k\mathbf{Q}_0^\top\left(\mathbf{Q}_0\breve{\mathbf{\Upsilon}}_k\mathbf{Q}_0^\top\right)^{-1}\mathbf{Q}_0\breve{\mathbf{\Upsilon}}_k\mathbf{Q}_L^\top = 0$ since its rank is null. Then, (10) becomes (6).

REFERENCES

[1] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the bayes tree," *Int. J. Robot. Res.*, vol. 31, no. 2, pp. 216–235, 2011.

[2] V. Ila, L. Polok, M. Solony, and P. Svoboda, "SLAM++—A highly efficient and temporally scalable incremental SLAM framework," *Int. J. Robot. Res.*, vol. 36, no. 2, pp. 210–230, 2017.

[3] H. Johannsson, M. Kaess, M. Fallon, and J. Leonard, "Temporally scalable visual SLAM using a reduced pose graph," in *Proc. IEEE Int. Conf. Robot. Autom.*, Karlsruhe, Germany, May 2013, pp. 54–61.

[4] L. Carlone and K. Sertac, "Attention and anticipation in fast visual-inertial navigation," in *Proc. Int. Conf. Robot. Autom.*, 2017, pp. 3886–3893.

[5] V. Ila, J. M. Porta, and J. Andrade-Cetto, "Information-based compact Pose SLAM," *IEEE Trans. Robot.*, vol. 26, no. 1, pp. 78–93, Feb. 2010.

[6] J. Vial, H. Durrant-Whyte, and T. Bailey, "Conservative sparsification for efficient and consistent approximate estimation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, San Francisco, CA, USA, Sep. 2011, pp. 886–893.

[7] H. Kretzschmar and C. Stachniss, "Information-theoretic compression of pose graphs for laser-based SLAM," *Int. J. Robot. Res.*, vol. 31, no. 11, pp. 1219–1230, 2012.

[8] S. Choudhary, V. Indelman, H. Christensen, and F. Dellaert, "Information-based reduced landmark SLAM," in *Proc. IEEE Int. Conf. Robot. Autom.*, Seattle, WA, USA, May 2015, pp. 4620–4627.

[9] K. Khosoussi, G. S. Sukhatme, S. Huang, and G. Dissanayake, "Designing sparse reliable pose-graph slam: A graph-theoretic approach," arXiv:1611.00889, 2016.

[10] N. Carlevaris-Bianco, M. Kaess, and R. M. Eustice, "Generic node removal for factor-graph SLAM," *IEEE Trans. Robot.*, vol. 30, no. 6, pp. 1371–1385, Dec. 2014.

[11] K. Eckenhoff, L. Paull, and G. Huang, "Decoupled, consistent node removal and edge sparsification for graph-based SLAM," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Daejeon, South Korea, pp. 3275–3282.

[12] M. Mazuran, W. Burgard, and G. D. Tipaldi, "Nonlinear factor recovery for long-term SLAM," *Int. J. Robot. Res.*, vol. 35, no. 1–3, pp. 50–72, 2016.

[13] J. Vallvé, J. Solà, and J. Andrade-Cetto, "Factor descent optimization for sparsification in graph SLAM," in *Proc. Eur. Conf. Mobile Robots*, Paris, France, Sep. 2017, pp. 95–100.

[14] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," in *Proc. Auton. Robot Veh.*, 1990, pp. 167–193.

[15] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g^{2}o: A general framework for graph optimization," in *Proc. IEEE Int. Conf. Robot. Autom.*, Shanghai, China, May 2011, pp. 3607–3613.

[16] F. Dellaert and M. Kaess, "Square root SAM: Simultaneous localization and mapping via square root information smoothing," *Int. J. Robot. Res.*, vol. 25, no. 12, pp. 1181–1204, 2006.

[17] M. Schmidt, E. Berg, M. Friedlander, and K. Murphy, "Optimizing costly functions with simple constraints: A limited-memory projected quasi-Newton algorithm," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2009, pp. 456–463.

[18] M. Mazuran, G. D. Tipaldi, L. Spinello, and W. Burgard, "Nonlinear graph sparsification for SLAM," in *Proc. Robot., Sci. Syst.*, Berkeley, CA, USA, Jul. 2014, pp. 1–8.

[19] S. Agarwal, K. Mierle, and Others, "Ceres solver," [Online]. Available: http://ceres-solver.org

[20] L. Carlone, [Online]. Available: http://www.lucacarlone.com/index.php/resources/datasets