Computational Enhancements for Certifiably Correct SLAM

David M. Rosen and Luca Carlone

***Abstract*—We investigate numerical and computational aspects of the use of convex relaxation for simultaneous localization and mapping (SLAM). Recent work has shown that convex relaxation provides an effective tool for computing, and certifying the correctness of, *globally optimal* SLAM solutions. This paper expands upon this prior work by demonstrating how to exploit the structure of the relaxed optimization problem to design very fast solvers, capable of computing globally optimal trajectories with thousands of poses in a fraction of a second. In particular, we describe several computational enhancements for accelerating the underlying Riemannian trust-region optimization method, including the use of structure-exploiting matrix decomposition, iterative linear-algebraic techniques, truncated-Newton methods, and preconditioning strategies. We also describe methods for accelerating the minimum-eigenvalue computation used to certify the optimality of a recovered estimate. We have incorporated these computational enhancements in an updated version of the *SE-Sync* library, and released the corresponding code online. Experimental results indicate that this enhanced implementation of SE-Sync is approximately twice as fast as GTSAM, a highly optimized, state-of-the-art library for SLAM.**

I. Introduction

The ability to accurately perceive the state of the world is a fundamental competency for autonomous systems, enabling such essential functions as planning, navigation, and manipulation [44]. However, solving this perceptual problem in practice presents a formidable challenge: perception algorithms must remain *robust and reliable* in the face of sensor noise, unmodeled dynamics, and perceptual ambiguity, while at the same time admitting *efficient computation* to support real-time operation of mobile robots with limited computational resources. In this work, we investigate the development of perceptual algorithms that satisfy *both* of these requirements, focusing in particular on the foundational problem of pose-graph *simultaneous localization and mapping* (SLAM) [42, 44].

Current state-of-the-art approaches to SLAM formulate the inference problem as a maximum-likelihood estimation (MLE) under an assumed probability distribution for the measurement noise [42]. This approach is attractive from a theoretical standpoint, due to the strong performance guarantees that maximum-likelihood estimation affords. However, the SLAM MLE is also *high-dimensional* and *nonconvex*, and therefore computationally hard to solve in general. In light of this, a major line of research in the SLAM literature has focused on the development of fast *approximate* inference methods (based upon smooth *local* optimization techniques) that can efficiently recover high-quality critical points of the SLAM MLE when supplied with a good initialization. This approach has enabled the development of remarkably performant algorithms and software libraries capable of processing SLAM problems involving tens to hundreds of thousands of poses in real time using only a single thread on a commodity processor [18, 23, 27, 29, 35, 36]. At the same time, however, this restriction to *local* search renders these methods vulnerable to convergence to significantly suboptimal critical points, even for relatively low levels of measurement noise [15, 35, 37].

To address this potential pitfall, a second line of SLAM research has studied statistical and numerical aspects of solving the SLAM MLE using smooth numerical optimization methods. Grisetti et al. [23], Olson et al. [35], and Tron et al. [46] present local optimization methods with larger basins of attraction to favorable critical points. Carlone and Censi [12], Carlone et al. [13] and Rosen et al. [37] propose initialization techniques to bootstrap local search. Carlone [11] and Rosen et al. [36] investigate convergence and numerical properties of smooth optimization methods applied to the SLAM MLE. Huang et al. [25], Wang et al. [47], and Khosoussi et al. [28] explore the global structure and statistical properties of the SLAM MLE itself. Finally, a very recent series of papers [14, 15, 38, 39] have developed the first *certifiably correct* SLAM algorithms; these are capable of *directly computing*, and *computationally certifying* the correctness of, *globally* optimal SLAM solutions, albeit at the cost of solving a large-scale semidefinite program (SDP) [45] and minimum-eigenvalue problem [22], respectively.

To date, these two lines of research (aimed at computational efficiency and global optimality, respectively) have often proceeded in parallel. The goal of this paper is to show that the mathematical insights gleaned from our prior work on certifiably correct SLAM [38, 39] can be exploited to design extremely fast *globally optimal* SLAM solvers, thereby achieving the best of both worlds. Concretely, we make the following contributions:

- We describe several design principles and computational enhancements that enabled us to develop a *fast*, *practical implementation* of *SE-Sync*, our certifiably correct SLAM algorithm [38, 39];
- We release a new C++ implementation of SE-Sync incorporating these computational enhancements;[1]
- We experimentally evaluate our C++ implementation of SE-Sync against GTSAM, a highly-optimized, state-of-the-art software library for SLAM.

Additionally, many of the general design principles and computational enhancements that we describe herein are quite broadly applicable; consequently, beyond its immediate util-

D.M. Rosen is with Oculus Research, Redmond, WA 98052, USA. Email: david.rosen@oculus.com

L. Carlone is with the Laboratory for Information & Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139, USA. Email: lcarlone@mit.edu

[1] Available at https://github.com/david-m-rosen/SE-Sync

ity for reporting the implementation details of the SE-Sync algorithm, it is our hope that this paper can also serve as a useful case study in the design and implementation of high-performance optimization techniques.

II. A review of certifiably correct SLAM

We begin by briefly reviewing the formulations of the pose-graph SLAM problem and its convex relaxation that form the basis of the SE-Sync algorithm. Interested readers are encouraged to consult [14, 38, 39] for additional details.

A. Pose-graph SLAM

In pose-graph SLAM, one estimates the values of a set of n unknown *poses* $x_1, \ldots, x_n \in \mathrm{SE}(d)$ given m noisy measurements of a subset of their pairwise relative transforms $x_{ij} \triangleq x_i^{-1} x_j$ [24, 42]. We model the set of available measurements using an undirected graph $G = (\mathcal{V}, \mathcal{E})$ in which the nodes $i \in \mathcal{V}$ are in one-to-one correspondence with the unknown poses x_i and the edges $\{i, j\} \in \mathcal{E}$ are in one-to-one correspondence with the set of available measurements, and assume without loss of generality that G is connected.[2] We let $\vec{G} = (\mathcal{V}, \vec{\mathcal{E}})$ be a directed graph obtained from G by fixing an orientation for each of its edges, and assume that a noisy measurement $\tilde{x}_{ij}$ of each relative pose $x_{ij} = (t_{ij}, R_{ij})$ is obtained by sampling from the following probabilistic generative model:[3]

$$\begin{aligned} \tilde{t}_{ij} &= \underline{t}_{ij} + t_{ij}^{\epsilon}, & t_{ij}^{\epsilon} &\sim \mathcal{N}\left(0, \tau_{ij}^{-1} I_d\right), \\ \tilde{R}_{ij} &= \underline{R}_{ij} R_{ij}^{\epsilon}, & R_{ij}^{\epsilon} &\sim \mathrm{Langevin}\left(I_d, \kappa_{ij}\right), \end{aligned} \quad \forall (i,j) \in \vec{\mathcal{E}}. \tag{1}$$

Here $\underline{x}_{ij} = (\underline{t}_{ij}, \underline{R}_{ij})$ is the true (latent) value of x_{ij}, $\mathcal{N}(\mu, \Sigma)$ denotes the standard multivariate Gaussian distribution with mean $\mu \in \mathbb{R}^d$ and covariance $\Sigma \succeq 0$, and $\mathrm{Langevin}(M, \kappa)$ denotes the *isotropic Langevin distribution* on $\mathrm{SO}(d)$ with mode $M \in \mathrm{SO}(d)$ and concentration parameter $\kappa \geq 0$ [6, 16].

Given a set of noisy measurements $\tilde{x}_{ij}$ sampled from the model (1), a straightforward computation (cf. [39, Sec. 2.2]) shows that a maximum-likelihood estimate $\hat{x}_{\mathrm{MLE}} \in \mathrm{SE}(d)^n$ for the states $x_1, \ldots, x_n$ is obtained as a minimizer of:

$$p^*_{\mathrm{MLE}} = \min_{\substack{t_i \in \mathbb{R}^d \\ R_i \in \mathrm{SO}(d)}} \sum_{(i,j) \in \vec{\mathcal{E}}} \left\{ \begin{array}{l} \kappa_{ij} \|R_j - R_i \tilde{R}_{ij}\|_F^2 \\ + \tau_{ij} \left\| t_j - t_i - R_i \tilde{t}_{ij} \right\|_2^2 \end{array} \right\}. \tag{2}$$

Moreover, we observe that fixing values for the orientations $R_1, \ldots, R_n \in \mathrm{SE}(d)$ in (2) reduces it to the *unconstrained* minimization of a convex quadratic form in the translational states $t_1, \ldots, t_n \in \mathbb{R}^d$, for which we can find a closed-form solution. This enables us to *analytically eliminate* the position estimates from (2), obtaining a simplification of the SLAM MLE involving only the orientations (cf. Appendices B.2 and B.3 in [39] for details):

$$p^*_{\mathrm{MLE}} = \min_{R \in \mathrm{SO}(d)^n} \mathrm{tr}(\tilde{Q} R^{\mathsf{T}} R) \tag{3a}$$

$$\tilde{Q} = L(\tilde{G}^{\rho}) + \tilde{T}^{\mathsf{T}} \Omega^{\frac{1}{2}} \Pi \Omega^{\frac{1}{2}} \tilde{T}, \tag{3b}$$

where here:

- $R \triangleq (R_1, \ldots, R_n) \in \mathrm{SO}(d)^n \subset \mathbb{R}^{d \times dn}$ is the block matrix obtained by concatenating the orientations R_i,
- $L(\tilde{G}^{\rho}) \in \mathrm{Sym}(dn)$ is the symmetric $(d \times d)$-block-structured *connection Laplacian* for the rotation-only estimation problem determined by the measurements $\tilde{R}_{ij}$ and measurement weights κ_{ij} for $(i,j) \in \vec{\mathcal{E}}$ [41]:

$$L(\tilde{G}^{\rho})_{ij} \triangleq \begin{cases} d_i^{\rho} I_d, & i = j, \\ -\kappa_{ij} \tilde{R}_{ij}, & \{i,j\} \in \mathcal{E}, \\ 0_{d \times d}, & \{i,j\} \notin \mathcal{E}, \end{cases} \tag{4a}$$

$$d_i^{\rho} \triangleq \sum_{e \in \delta(i)} \kappa_e, \tag{4b}$$

- $\tilde{T} \in \mathbb{R}^{m \times dn}$ is the $(1 \times d)$-block-structured translational data matrix with rows and columns indexed by $e \in \vec{\mathcal{E}}$ and $k \in \mathcal{V}$, respectively, and with (e,k)-block given by:

$$\tilde{T}_{ek} \triangleq \begin{cases} -\tilde{t}_{kj}^{\mathsf{T}}, & e = (k,j) \in \vec{\mathcal{E}}, \\ 0_{1 \times d}, & \text{otherwise}, \end{cases} \tag{5}$$

- $\Omega \triangleq \mathrm{Diag}(\tau_{e_1}, \ldots, \tau_{e_m}) \in \mathrm{Sym}(m)$ is a diagonal matrix of translational measurement precisions,
- $\Pi \in \mathbb{R}^{m \times m}$ is the matrix of the orthogonal projection operator $\pi \colon \mathbb{R}^m \to \ker(A(\vec{G})\Omega^{\frac{1}{2}})$ onto the kernel of the weighted *incidence matrix* $A(\vec{G})\Omega^{\frac{1}{2}}$ of $\vec{G}$ [4].

B. A convex relaxation for SLAM

The simplified pose-graph SLAM MLE (3) is a high-dimensional, nonconvex nonlinear program that is computationally hard to solve in general. To overcome this difficulty, SE-Sync employs *convex relaxation*; in this approach, one modifies a difficult optimization problem to obtain a (tractable) *convex approximation*, and then exploits this surrogate to search for good solutions of the original (hard) problem.

One common approach to generating such convex relaxations is to employ *Lagrangian duality* [8, Chp. 5]. In the specific case of the SLAM MLE (3), this strategy can be used to derive a convex relaxation in the form of a *semidefinite program* (cf. [39, Sec. 3.2] and [14, 15] for details):

$$\begin{aligned} p^*_{\mathrm{SDP}} &= \min_{Z \in \mathrm{Sym}(dn)} \mathrm{tr}(\tilde{Q} Z) \\ \text{s.t. } Z &= \begin{pmatrix} I_d & * & * & \cdots & * \\ * & I_d & * & \cdots & * \\ * & * & I_d & & * \\ \vdots & \vdots & & \ddots & \vdots \\ * & * & * & \cdots & I_d \end{pmatrix} \succeq 0. \end{aligned} \tag{6}$$

[2] If G is not connected, then the problem of estimating the unknown poses $x_1, \ldots, x_n$ decomposes into a set of independent estimation problems that are in one-to-one correspondence with the connected components of G; thus, the general case is always reducible to the case of connected graphs.

[3] We use a directed graph to model the measurements $\tilde{x}_{ij}$ sampled from (1) because the distribution of the noise corrupting the latent values $\underline{x}_{ij}$ is not invariant under $\mathrm{SE}(d)$'s group inverse operation. Consequently, we must keep track of which state x_i was the "base frame" for each measurement.

Comparing (3) and (6), we observe that the decision variable $R \in \mathrm{SO}(d)^n$ enters the objective in (3) only through the product $R^{\mathsf{T}}R$, which is a positive semidefinite matrix whose $(d \times d)$-block-diagonal is comprised of identity matrices, and is therefore feasible in (6); consequently, (6) can be regarded as a relaxation of the SLAM MLE obtained by *expanding* (3)*'s feasible set*. This immediately implies that $p^*_{\mathrm{SDP}} \leq p^*_{\mathrm{MLE}}$; i.e., the optimal value of (6) *lower-bounds* that of (3). Moreover, if it so happens that a minimizer Z^* of (6) admits a decomposition of the form $Z^* = {R^*}^{\mathsf{T}} R^*$ for some $R^* \in \mathrm{SO}(d)^n$, it is straightforward to verify that this R^* is also a minimizer of (3). The crucial fact that justifies our interest in the relaxation (6) is that it frequently *does* have a minimizer Z^* of just this form. More precisely, we have the following:

Proposition 1 (Proposition 1 of [38]): Let $\underline{Q}$ be the matrix of the form (3b) constructed using the true relative transforms $\underline{x}_{ij} = (\underline{t}_{ij}, \underline{R}_{ij})$ in (1). There exists a constant $\beta \triangleq \beta(\underline{Q}) > 0$ (depending upon $\underline{Q}$) such that, if $\|\tilde{Q} - \underline{Q}\|_2 < \beta$, then:

(i) The relaxation (6) has a unique solution Z^*, and

(ii) $Z^* = {R^*}^{\mathsf{T}} R^*$, where $R^* \in \mathrm{SO}(d)^n$ is a (global) minimizer of the simplified SLAM MLE (3).

In short, Proposition 1 guarantees that so long as the noise corrupting the measurements $\tilde{x}_{ij}$ in (1) is not too large (as measured by the spectral norm of the deviation of the data matrix $\tilde{Q}$ from its ground truth value $\underline{Q}$),[4] we can recover a *global* minimizer R^* of the SLAM MLE (3) by solving the convex relaxation (6) using *any* numerical method.

III. Computational enhancements for certifiably correct SLAM

As a semidefinite program, (6) can in principle be solved in polynomial time using interior-point methods [45]. In practice, however, the high per-iteration computational cost of general-purpose semidefinite programming algorithms prevents these methods from scaling effectively to problems in which the dimension of the decision variable Z is greater than a few thousand [45]. Unfortunately, typical instances of (6) are one to two orders of magnitude larger than this maximum effective problem size, and are therefore well beyond the reach of these general-purpose techniques. To overcome this limitation, SE-Sync employs a specialized optimization method specifically designed to solve large-scale instances of (6) efficiently. In this section, we summarize the general design principles and computational enhancements that we employed in order to implement this custom optimization procedure.

A. Exploiting problem structure

In general, the successful design of a performant optimization algorithm crucially depends upon the effective identification and exploitation of problem structure. Consequently, we begin the design of our optimization procedure by investigating the structure of problem (6) more deeply.

1) Exploiting low-rank structure: The dominant computational cost in applying general-purpose semidefinite programming methods to solve (6) is the need to store and manipulate expressions involving the dense $dn \times dn$ matrix variable Z; in particular, the $O(n^3)$ computational cost of multiplying and factoring such matrices quickly becomes intractable as the problem size n increases.

On the other hand, in the (typical) case that the hypotheses of Proposition 1 are satisfied we know that the actual *solution* Z^* of (6) that we seek has a very concise description in the factored form $Z^* = {R^*}^{\mathsf{T}} R^*$ for $R^* \in \mathrm{SO}(d)^n$. More generally, even in those cases where the hypotheses of Proposition 1 do *not* hold, minimizers Z^* of (6) still typically have a rank r not much greater than d, and therefore admit a symmetric rank decomposition $Z^* = {Y^*}^{\mathsf{T}} Y^*$ for $Y^* \in \mathbb{R}^{r \times dn}$ with $r \ll dn$.

Burer and Monteiro [9, 10] proposed an elegant approach to exploit the fact that large-scale semidefinite programs often admit such low-rank solutions: simply replace every instance of the decision variable Z with a rank-r product of the form $Y^{\mathsf{T}}Y$ to produce a *rank-restricted* version of the original problem. This substitution has the two-fold effect of (i) dramatically reducing the size of the search space and (ii) rendering the positive semidefiniteness constraint *redundant*, since $Y^{\mathsf{T}}Y \succeq 0$ for *any* choice of Y. The resulting rank-restricted form of the problem is thus a low-dimensional *nonlinear* program, rather than a *semidefinite* program.

2) Exploiting geometric structure: Furthermore, following Boumal [5], we observe that after replacing Z with $Y^{\mathsf{T}}Y$ for $Y \triangleq (Y_1, \ldots, Y_n) \in \mathbb{R}^{r \times dn}$, the block-diagonal constraints in (6) are equivalent to $Y_i^{\mathsf{T}} Y_i = I_d$, i.e., the columns of each block $Y_i \in \mathbb{R}^{r \times d}$ form an *orthonormal frame*. In general, the set of all orthonormal k-frames in $\mathbb{R}^p$ $(k \leq p)$:

$$\mathrm{St}(k, p) \triangleq \{Y \in \mathbb{R}^{p \times k} \mid Y^{\mathsf{T}} Y = I_k\} \tag{7}$$

forms a smooth compact matrix manifold, called the *Stiefel manifold*, which can be equipped with a Riemannian metric induced by its embedding into $\mathbb{R}^{p \times k}$ [2, Sec. 3.3.2]. Together, these observations enable us to reduce (6) to an *unconstrained*, low-dimensional *rank-restricted* optimization problem defined on a product of Stiefel manifolds:

$$p^*_{\mathrm{SDPLR}} = \min_{Y \in \mathrm{St}(d,r)^n} \mathrm{tr}(\tilde{Q} Y^{\mathsf{T}} Y). \tag{8}$$

This is the optimization problem that we will actually solve.

3) Exploiting graph-theoretic structure: Simplifying (6) to (8) obviates the need to manipulate the large, dense matrix variable Z. However the data matrix $\tilde{Q}$ appearing in problems (3), (6), and (8) is also dense and of the same order as Z, and so presents a similar computational difficulty. Accordingly, here we develop an analogous concise description of $\tilde{Q}$.

The density of $\tilde{Q}$ is due to the density of the orthogonal projection matrix Π in (3b); since this matrix is ultimately derived from the structure of a sparse graph, we might suspect

[4] While Proposition 1 is formulated as a simple existence result for β, as a practical matter we have shown in our previous work [14, 38, 39] that the relaxation (6) remains exact when the available measurements $\tilde{x}_{ij}$ in (1) are corrupted by noise up to an order of magnitude greater than what is typically encountered in robotics and computer vision applications; see e.g. the discussion in footnote 11 of [39].

that it too should admit some kind of sparse description. And indeed, it turns out that Π admits a sparse decomposition as:

$$\underline{A}(\vec{G})\Omega^{\frac{1}{2}} = LQ_1 \tag{9a}$$

$$\Pi = I_m - \Omega^{\frac{1}{2}}\underline{A}(\vec{G})^{\mathsf{T}}L^{-\mathsf{T}}L^{-1}\underline{A}(\vec{G})\Omega^{\frac{1}{2}} \tag{9b}$$

where equation (9a) is a thin LQ decomposition of the weighted reduced incidence matrix $\underline{A}(\vec{G})\Omega^{\frac{1}{2}}$ of $\vec{G}$; this result is derived in [39, Appendix B.3]. Note that (9b) requires only the sparse lower-triangular factor L from (9a), which can be efficiently obtained using e.g. Givens rotations [22, Sec. 5.2.1].

4) Ensuring optimality: While the reduction from (6) to (8) dramatically reduces the size of the search space, it comes at the expense of (re)introducing the (nonconvex) quadratic orthonormality constraints in (7). It may therefore not be clear whether anything has really been gained, since it appears that we may have simply replaced one difficult nonconvex optimization problem with another. The following remarkable result (adapted from Boumal et al. [7]) justifies this approach:

Proposition 2 (Proposition 2 of [38]): If $Y^* \in \mathrm{St}(d,r)^n$ is a (row) rank-deficient second-order critical point of (8), then Y^* is a minimizer, and $Z^* = Y^{*\mathsf{T}}Y^*$ is a solution of (6).

Proposition 2 immediately suggests a procedure for obtaining solutions Z^* of (6) by applying a Riemannian optimization method to search successively higher levels r of the hierarchy of rank-restricted relaxations (8) until a rank-deficient second-order critical point is found.[5] This algorithm is referred to as the *Riemannian Staircase* [5, 7].

In light of Proposition 2, the SE-Sync algorithm implements two main computational procedures:

- A fast Riemannian trust-region method, to rapidly identify *first-order* critical points $Y^* \in \mathrm{St}(d,r)^n$ of (8),
- A fast minimum-eigenvalue computation, to determine whether $\operatorname{Hess} F(Y^*) \succeq 0$, and thereby establish the *global* optimality of Y^*.[6]

The next two subsections describe the design of efficient algorithms for each of these.

B. Designing a fast Riemannian optimization method for (8)

The basic model for superlinear smooth minimization algorithms on Riemannian manifolds is *Newton's method* [1]: given a twice-continuously differentiable function $f\colon \mathcal{M} \to \mathbb{R}$ on a Riemannian manifold $\mathcal{M}$ and an initial estimate $x_k \in \mathcal{M}$ for a minimizer x^* of f, we construct a quadratic model function q_{x_k} for f on the tangent space $T_{x_k}\mathcal{M}$ of $\mathcal{M}$ at x_k:

$$\begin{aligned} q_{x_k}&\colon T_{x_k}\mathcal{M} \to \mathbb{R} \\ q_{x_k}(\eta) &= f(x_k) + g_{x_k}(\operatorname{grad} f(x_k), \eta) + \frac{1}{2} g_{x_k}(H_{x_k}\eta, \eta) \end{aligned} \tag{10}$$

(where $g_{x_k}\colon T_{x_k}\mathcal{M} \times T_{x_k}\mathcal{M} \to \mathbb{R}$ is the (Riemannian) inner product on $T_{x_k}\mathcal{M}$ and $H_{x_k}\colon T_{x_k}\mathcal{M} \to T_{x_k}\mathcal{M}$ is a symmetric approximation of the Riemannian Hessian), compute an *update step* $\eta_k \in T_{x_k}\mathcal{M}$ by (possibly approximately) solving the *trust-region subproblem*:

$$\begin{aligned} \eta_k \in \operatorname*{argmin}_{\eta \in T_{x_k}\mathcal{M}} & q_{x_k}(\eta) \\ \text{s.t. } & g_{x_k}(\eta, \eta) \leq \Delta_k^2, \end{aligned} \tag{11}$$

and then obtain an improved estimate x_{k+1} by applying a *retraction* [1, Def. 2.1] $R_{x_k}\colon T_{x_k}\mathcal{M} \to \mathcal{M}$ to move along the manifold in the direction of the trust-region update step η_k:

$$x_{k+1} = R_{x_k}(\eta_k). \tag{12}$$

Designing a Riemannian optimization method within this class thus reduces to selecting specific procedures for (i) constructing the Hessian approximation H_{x_k} in (10), (ii) solving the trust-region subproblem in (11), and (iii) retracting along tangent vectors as in (12).

1) A closed form for the Riemannian gradient and Hessian: We begin the design of our optimization method by first determining closed-form expressions for the Riemannian gradient and the action of the Riemannian Hessian of the objective $F(Y) \triangleq \operatorname{tr}(\tilde{Q}Y^{\mathsf{T}}Y)$ of (8). Considering $F(Y)$ as a function on the ambient Euclidean space $\mathbb{R}^{r\times dn}$, a straightforward computation shows that its gradient and Hessian operator are:

$$\nabla F(Y) = 2Y\tilde{Q}, \quad \nabla^2 F(Y)[\dot{Y}] = 2\dot{Y}\tilde{Q}. \tag{13}$$

Furthermore, there are simple relations between the ambient Euclidean gradient and Hessian operator in (13) and their Riemannian counterparts when F is viewed as a function restricted to the embedded submanifold $\mathrm{St}(d,r)^n \subset \mathbb{R}^{r\times dn}$ as in (8) (cf. eqs. (3.37) and (5.15) and of [2]):

$$\begin{aligned} \operatorname{grad} F(Y) &= \operatorname{Proj}_Y \nabla F(Y) \\ \operatorname{Hess} F(Y)[\dot{Y}] &= \operatorname{Proj}_Y\left(\nabla^2 F(Y)[\dot{Y}]\right) \\ &\quad - \operatorname{Proj}_Y\left(\dot{Y}\operatorname{SymBlockDiag}_d\left(Y^{\mathsf{T}}\nabla F(Y)\right)\right), \end{aligned} \tag{14}$$

where Proj_Y denotes the orthogonal projection operator onto the tangent space of $\mathrm{St}(d,r)^n$ at Y [21, eq. (2.3)]:

$$\begin{aligned} \operatorname{Proj}_Y&\colon T_Y\left(\mathbb{R}^{r\times dn}\right) \to T_Y\left(\mathrm{St}(d,r)^n\right) \\ \operatorname{Proj}_Y(X) &= X - Y\operatorname{SymBlockDiag}_d(Y^{\mathsf{T}}X) \end{aligned} \tag{15}$$

and $\operatorname{SymBlockDiag}_d\colon \mathbb{R}^{dn\times dn} \to \mathbb{R}^{dn\times dn}$ maps a $dn \times dn$ matrix M to a $(d\times d)$-block-diagonal matrix whose diagonal blocks are the symmetrizations of M's [39, eqs. (4) & (5)].

Together, equations (3b), (9), and (13)–(15) provide an efficient means of evaluating the objective $F(Y)$, its Riemannian gradient $\operatorname{grad} F(Y)$, and the action of the Riemannian Hessian $\operatorname{Hess} F(Y)[\dot{Y}]$ on a tangent vector $\dot{Y}$, via a sequence of sparse matrix multiplications and sparse triangular solves.

[5] Note that since *every* $Y \in \mathrm{St}(d,r)^n$ is row rank-deficient for $r > dn$, this algorithm is guaranteed to recover an optimal solution after searching at most $dn + 1$ levels of the hierarchy (8).

[6] In the event that $\operatorname{Hess} F(Y^*) \not\succeq 0$, the eigenvector corresponding to the (negative) minimum eigenvalue of $\operatorname{Hess} F(Y^*)$ provides a descent direction from Y^* that is used to initialize the optimization for the next "stair" (8) in the Riemannian Staircase, cf. Theorem 3.9 and Corollary 3.10 of [5].

2) Solving the trust-region subproblem: It is a standard result that the trust-region step η_k defined by (11) can be obtained by solving a linear equation of the form:

$$(H_{x_k} + \lambda W_{x_k})\eta_k = -\operatorname{grad} f(x_k), \tag{16}$$

where $\lambda \geq 0$ is a Lagrange multiplier associated with the trust-region constraint in (11) and $W_{x_k} \succ 0$ is a weight matrix associated with the inner product g_{x_k}.[7]

If the coefficient matrix $H_{x_k} + \lambda W_{x_k}$ can be readily factored (using e.g. QR or Cholesky decomposition [22]), (16) can be solved accurately and efficiently by exploiting this decomposition. In the context of SLAM specifically, this approach has become the *de facto* standard since the seminal work of Dellaert and Kaess [17], and forms the basis of current state-of-the-art superlinear optimization methods for solving (2) [27, 29, 36]. However, in the specific case of problem (8), equations (3b) and (13)–(15) imply that the Hessian appearing in (16) will be *high-rank* and *dense*, which renders the use of direct factorization techniques infeasible.

Alternatively, one can also solve (16) using iterative numerical methods [22, Chp. 10]. These techniques have the desirable property that they do not require direct access to the coefficient matrix itself; rather, they require only the ability to compute *products* with this matrix. As a result, iterative approaches can realize significant computational and memory savings versus direct factorization methods whenever efficient procedures for evaluating such matrix-vector products are available.

Furthermore, iterative linear solvers provide an additional opportunity for computational acceleration when used to solve (11) (via (16)) specifically in the context of nonlinear optimization. Recall that the trust-region step η_k is obtained by minimizing the *model* function q_{x_k}, rather than f itself; as a result, computing a higher-accuracy estimate of η_k does not necessarily guarantee a corresponding improvement in the reduction of f achieved when η_k is applied. This observation motivates the design of *inexact Newton methods* [20]; rather than solving each instance of (11) *exactly*, these techniques merely *approximate* η_k to within a specified tolerance. By dynamically adjusting this tolerance as they run, inexact Newton methods can avoid the computational expense of *oversolving* the trust-region subproblems (especially in early iterations, where the current iterate x_k may be far from the optimum x^*, and only a relatively coarse estimate of η_k is required to make good progress) while still guaranteeing a superlinear convergence rate and high-accuracy estimation of x^*. Iterative linear solvers provide a particularly elegant means of performing this controllable approximation of (11), as they can simply be *terminated* as soon as any iterate achieves the required precision. This specific approach forms the basis of *truncated-Newton methods* [19, 43], which comprise the current state of the art for superlinear large-scale unconstrained nonlinear programming [34, Sec. 7.1].

[7] This follows from a straightforward Lagrangian analysis of the trust-region subproblem; cf. e.g. [2, Sec. 7.3.1].

In light of these considerations, we make the following concrete selections for each of the three major elements of our Riemannian optimization method:

(i) **Hessian approximation:** We use the *exact* Hessian;
(ii) **Trust-region subproblem solver:** We employ a Riemannian adaptation [1] of Steihaug's truncated conjugate-gradient method [43] to solve the trust-region subproblems (11) efficiently, using the method described in Section III-B1 to evaluate Hessian-vector products;
(iii) **Retraction:** We use the thin QR decomposition-based *Q-factor* retraction described in equation (4.8) of [2].

3) Preconditioning the conjugate gradient method: As a novel contribution of this work (not present in the original formulation of the SE-Sync algorithm [38, 39]), we show how to further accelerate the solution of the trust-region subproblems (11) by *preconditioning* Steihaug's truncated conjugate-gradient method.

Recall that the convergence rate of the conjugate gradient method when solving a linear system $Ax = b$ is controlled by the condition number $\kappa(A)$ of A [22, Sec. 10.2.7]. As a result, it is possible to dramatically enhance the convergence of the method by solving the (equivalent) linear system $PAx = Pb$, where $P \succ 0$ is chosen such that $\kappa(PA) \ll \kappa(A)$. Of course, the ideal preconditioner is then trivially $P = A^{-1}$, for which $\kappa(PA) = 1$; however, *constructing* this P then amounts to solving the original linear system, and so provides no *computational* advantage. Designing an effective preconditioner P thus amounts to striking a suitable balance between the improvement in the conditioning of the linear system obtained (enabling the conjugate gradient iterations to *converge* rapidly) and the computational cost of constructing and applying the preconditioning operator P (so that each conjugate gradient iteration can be *computed* rapidly).

In the specific case of (11), the matrix $\tilde{Q}$ from (3b) that determines the Riemannian Hessian operator (13)–(15) is obtained as the (generalized) Schur complement of $L(W^\tau)$ in the 2×2 block matrix [39, eq. (18)]:

$$\begin{pmatrix} L(W^\tau) & \tilde{V} \\ \tilde{V}^\mathsf{T} & L(\tilde{G}^\rho) + \tilde{\Sigma} \end{pmatrix} \tag{17}$$

(see [39, Appendix B.1] for this derivation); this suggests that a preconditioner designed for $L(\tilde{G}^\rho) + \tilde{\Sigma}$ will also be effective at preconditioning $\tilde{Q}$ itself (cf. e.g. Theorem 4.2 of [31]). Since $\tilde{\Sigma}$ is a dense $(d \times d)$-block-diagonal modification of $L(\tilde{G}^\rho)$, we make the additional simplifying decision to design a preconditioner using $L(\tilde{G}^\rho)$ alone, both to encourage additional sparsity and to take advantage of the close connection between $L(\tilde{G}^\rho)$ and the structure of the underlying graph $\tilde{G}^\rho$.

Based on these considerations, and in light of (13)–(14), we propose to use a preconditioning operator of the form:

$$P(\dot{Y}) = \frac{1}{2}\operatorname{Proj}_Y\left(\dot{Y}M^{-1}\right) \tag{18}$$

with either $M = \operatorname{SymBlockDiag}_d(L(\tilde{G}^\rho))$ (i.e. *block-Jacobi* preconditioning) or $M = CC^\mathsf{T}$, where C is an *incomplete*

Cholesky factorization [30] of $L(\tilde{G}^\rho)$.[8]

C. Accelerated optimality verification

The Riemannian optimization method of Section III-B enables us to quickly recover a *first-order* critical point Y^* of (8); given such a point, our next objective is then to determine whether Y^* is *second-order* (and hence globally optimal for (8) and (6) according to Proposition 2).

Using a Lagrangian analysis of (8), one can show that a critical point Y^* is second-order if and only if the matrix:

$$C(Y^*) \triangleq \tilde{Q} - \mathrm{SymBlockDiag}_d(\tilde{Q}Y^{*\mathsf{T}}Y^*) \qquad (19)$$

is positive semidefinite (cf. Theorems 3.7, 3.8, and 3.9 of [5]); consequently, our task reduces to computing the minimum eigenvalue of $C(Y^*)$. As in the case of Section III-B, the presence of the (large, dense) matrix $\tilde{Q}$ in (19) renders direct factorization-based approaches infeasible, so we will again employ iterative linear-algebraic methods.

The *Lanczos algorithm* [22, Chp. 9] is an iterative numerical linear-algebraic procedure for computing specific eigenvalue-eigenvector pairs (λ, v) of a symmetric linear operator A. In brief, when initialized with a vector l_0, this method constructs the *Krylov subspace* $K \triangleq \mathrm{span}\{l_0, Al_0, \ldots, A^{k-1}l_0\}$, computes $A' \triangleq \pi_K A|_K$ (the orthogonal projection of the operator A onto K), and then approximates the target eigenvalue-eigenvector pair (λ, v) of A using the best available eigenpair (θ, y) of A'. Unsurprisingly, two of the critical factors influencing the convergence rate of this method are (i) the location of the target eigenvalue λ within the spectrum of A and (ii) the proximity of the initial vector l_0 to the eigenspace associated with the target eigenvalue λ [40]. In this section, we show how to construct a minimum-eigenvalue estimation procedure for (19) that improves both of these factors.

1) Spectrum shifting: Let $\lambda_{\min}$ denote the minimum eigenvalue of $C(Y^*)$. We describe a procedure for determining $\lambda_{\min}$ that requires only computation of *dominant* (i.e. *maximum-magnitude*) eigenvalues; these typically have very favorable convergence properties.

We begin by computing λ_{dom}, the dominant eigenvalue of $C(Y^*)$; this can be done reasonably efficiently starting from a random initial vector and applying a few Lanczos or power iterations [22, Sec. 8.2]. If $\lambda_{\mathrm{dom}} < 0$, then $\lambda_{\min} = \lambda_{\mathrm{dom}}$ and we are done. Otherwise, $\lambda_{\mathrm{dom}} \geq 0$, and we consider the operator:

$$S(Y^*) \triangleq C(Y^*) - \lambda_{\mathrm{dom}} I. \qquad (20)$$

A simple diagonalization argument shows that $S(Y^*)$ and $C(Y^*)$ have the same set of eigenvectors, and that the eigenvalues λ_i and θ_i of $C(Y^*)$ and $S(Y^*)$ (respectively) are related by $\theta_i = \lambda_i - \lambda_{\mathrm{dom}}$; in particular, since $\lambda_i - \lambda_{\mathrm{dom}} < 0$ for all i, it follows that the dominant eigenvalue of $S(Y^*)$ is $\theta_{\mathrm{dom}} = \lambda_{\min} - \lambda_{\mathrm{dom}}$. Consequently, we can recover $\lambda_{\min}$ by computing θ_{dom} using the Lanczos procedure applied to $S(Y^*)$, and then calculate $\lambda_{\min} = \theta_{\mathrm{dom}} + \lambda_{\mathrm{dom}}$.

[8]We also experimented with the use of maximum-weight spanning tree [3] and L-BFGS preconditioners [33], but found that these were not as effective as either straightforward block-Jacobi or incomplete Cholesky preconditioning.

2) Initialization: In addition to spectral transformation, we can also exploit the geometry of (8) to determine a favorable initialization l_0 for the Lanczos iterations.

A straightforward computation shows that the Lagrangian form of the first-order necessary condition for (8) is

$$C(Y^*)Y^{*\mathsf{T}} = 0; \qquad (21)$$

i.e., *the rows of Y^* are always eigenvectors of $C(Y^*)$ with eigenvalue* 0. In the (typical) case that the optimization method of Section III-B converges to a global minimizer, the fact that $C(Y^*) \succeq 0$ and has 0 as an eigenvalue implies that $\lambda_{\min} = 0$, and therefore the rows of Y^* directly provide us a set of eigenvectors associated with the minimum eigenvalue!

This observation might suggest initializing the Lanczos procedure with one of the rows of Y^*, say y_1. However, in the event that $C(Y^*) \not\succeq 0$ and we use y_1 as an initial vector, the fact that y_1 is already a pure eigenvector means that the Lanczos procedure will never escape the (invariant) subspace spanned by y_1. A better initialization is to perturb y_1 to $\tilde{y}_1$ by adding a small quantity of random noise; this enables $\tilde{y}_1$ to remain close to the 0-eigenspace (enabling very rapid convergence in the event that $C(Y^*) \succeq 0$) while still retaining components from *all* of $C(Y^*)$'s eigenspaces. We have found that generating $\tilde{y}_1$ by adding a spherically-uniformly distributed random vector of magnitude $.03 \cdot \|y_1\|_2$ to y_1 works very well in practice.

IV. Experimental results

As a main contribution of this work, we have publicly released a C++ implementation of the SE-Sync algorithm that incorporates the computational enhancements detailed in Section III. Our code employs the Riemannian Trust-Region (RTR) [1] implementation from ROPTLIB [26] to solve the rank-restricted Riemannian optimization (8), and the Lanczos implementation from the Spectra library[9] to perform the minimum-eigenvalue computation of Section III-C. In this section, we compare this C++ implementation of SE-Sync against the Gauss-Newton method implemented in GTSAM,[10] a highly-optimized, state-of-the-art software library specifically designed for SLAM applications.

Our experimental test set consists of the six large-scale SLAM benchmark datasets considered in our previous work [38, 39]; the first three of these (the *sphere*, *torus*, and *grid* datasets) are synthetic (although generated using an observation model different from (1)), while the latter three (the *garage*, *cubicle*, and *rim* datasets) are large-scale real-world examples. All of the following experiments are performed on a Dell Precision 5510 laptop with an Intel Xeon E3-1505M 2.80 GHz processor and 16 GB of RAM running Ubuntu 16.04. We initialize each solver using the *chordal initialization*, a state-of-the-art method for bootstrapping an initial solution in SLAM and bundle adjustment problems [32]. Each algorithm is limited to a maximum of 300 iterations, and

[9]Available at https://github.com/yixuan/spectra

[10]Version 4.0, available at https://bitbucket.org/gtborg/gtsam

			GTSAM		SE-Sync			
	# Poses	# Measurements	Objective value	99% red. time [s]	Objective value	99% red. time [s]	Global opt.?	Verification time [s]
sphere	2500	4949	1.687×10^3	0.444	1.687×10^3	0.149	✓	0.245
torus	5000	9048	2.423×10^4	0.840	2.423×10^4	0.614	✓	0.538
grid	8000	22236	8.432×10^4	29.79	8.432×10^4	1.301	✓	1.643
garage	1661	6275	1.263×10^0	0.278	1.263×10^0	0.250	✓	6.456
cubicle	5750	16869	7.171×10^2	1.176	7.171×10^2	0.700	✓	1.722
rim	10195	29743	5.461×10^3	4.969	5.461×10^3	1.900	✓	11.68

TABLE I
RESULTS FOR THE SLAM BENCHMARK DATASETS.

the RTR method is limited to a maximum of 500 Hessian-vector products per outer iteration. The two algorithms make use of slightly different convergence criteria: SE-Sync employs a criterion based upon the norm of the Riemannian gradient, whereas GTSAM uses a relative decrease condition; in these experiments, we set these parameters to 10^{-2} and 10^{-5}, respectively. To enable a fair comparison of computational speed despite the methods' use of different stopping criteria, we report the elapsed time necessary for them to achieve 99% of the total reduction in function value (both methods converged to the same solution in all of our examples). In the case of SE-Sync, we also report the time necessary to perform the global optimality verification, and consider the minimum eigenvalue to be numerically nonnegative if $\lambda_{\min} > -10^{-6}$. Results for these experiments are summarized in Table I.

Overall, these results indicate that the computational enhancements described in Section III enable SE-Sync to solve pose-graph SLAM problems approximately twice as fast as an existing state-of-the-art technique, while additionally *computationally certifying* the optimality of the solutions so recovered.

V. CONCLUSION

This paper describes the primary design principles and computational enhancements that we employed in order to obtain an efficient C++ implementation of SE-Sync, our convex relaxation-based algorithm for certifiably correct SLAM [38, 39]. Experimental results indicate that this accelerated implementation is capable of achieving state-of-the-art computational speeds on pose-graph SLAM problems, while additionally enabling the direct recovery of *certifiably globally optimal* solutions.

REFERENCES

[1] P.-A. Absil, C.G. Baker, and K.A. Gallivan. Trust-region methods on Riemannian manifolds. *Found. Comput. Math.*, 7(3):303–330, July 2007.

[2] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2008.

[3] M. Bern, J.R. Gilbert, B. Hendrickson, N. Nguyen, and S. Toledo. Support-graph preconditioners. *SIAM J. Matrix Anal. Appl.*, 27(4):930–951, 2006.

[4] N. Biggs. Algebraic potential theory on graphs. *Bull. London Math. Soc.*, 29:641–682, 1997.

[5] N. Boumal. A Riemannian low-rank method for optimization over semidefinite matrices with block-diagonal constraints. arXiv preprint: arXiv:1506.00575v2, 2015.

[6] N. Boumal, A. Singer, P.-A. Absil, and V.D. Blondel. Cramér-Rao bounds for synchronization of rotations. *Information and Inference*, 3:1–39, 2014.

[7] N. Boumal, V. Voroninski, and A.S. Bandeira. The non-convex Burer-Monteiro approach works on smooth semidefinite programs. arXiv preprint arXiv:1606.04970v1, June 2016.

[8] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[9] S. Burer and R.D.C. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Math. Program.*, 95:329–357, 2003.

[10] S. Burer and R.D.C. Monteiro. Local minima and convergence in low-rank semidefinite programming. *Math. Program.*, 103:427–444, 2005.

[11] L. Carlone. A convergence analysis for pose graph optimization via Gauss-Newton methods. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 965–972, Karlsruhe, Germany, May 2013.

[12] L. Carlone and A. Censi. From angular manifolds to the integer lattice: Guaranteed orientation estimation with application to pose graph optimization. *IEEE Trans. on Robotics*, 30(2):475–492, April 2014.

[13] L. Carlone, R. Aragues, J. Castellanos, and B. Bona. A fast and accurate approximation for planar pose graph optimization. *Intl. J. of Robotics Research*, 33(7):965–987, 2014.

[14] L. Carlone, D.M. Rosen, G. Calafiore, J.J. Leonard, and F. Dellaert. Lagrangian duality in 3D SLAM: Verification techniques and optimal solutions. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, September 2015.

[15] L. Carlone, G.C. Calafiore, C. Tommolillo, and F. Dellaert. Planar pose graph optimization: Duality, optimal solutions, and verification. *IEEE Trans. on Robotics*, 32 (3):545–565, June 2016.

[16] A. Chiuso, G. Picci, and S. Soatto. Wide-sense estimation on the special orthogonal group. *Communications in Information and Systems*, 8(3):185–200, 2008.

[17] F. Dellaert and M. Kaess. Square Root SAM: Simultaneous localization and mapping via square root information smoothing. *Intl. J. of Robotics Research*, 25(12):1181–

1203, December 2006.

[18] F. Dellaert, J. Carlson, V. Ila, K. Ni, and C.E. Thorpe. Subgraph-preconditioned conjugate gradients for large scale SLAM. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 2566–2571, Taipei, Taiwan, October 2010.

[19] R.S. Dembo and T. Steihaug. Truncated-Newton algorithms for large-scale unconstrained optimization. *Math. Program.*, 26:190–212, 1983.

[20] R.S. Dembo, S.C. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM J. Numer. Anal.*, 19(2):400–408, April 1982.

[21] A. Edelman, T.A. Arias, and S.T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM J. Matrix Anal. Appl.*, 20(2):303–353, October 1998.

[22] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 3rd edition, 1996.

[23] G. Grisetti, C. Stachniss, and W. Burgard. Nonlinear constraint network optimization for efficient map learning. *IEEE Trans. on Intelligent Transportation Systems*, 10(3):428–439, September 2009.

[24] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard. A tutorial on graph-based SLAM. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.

[25] S. Huang, Y. Lai, U. Frese, and G. Dissanayake. How far is SLAM from a linear least squares problem? In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 3011–3016, Taipei, Taiwan, October 2010.

[26] W. Huang, P.-A. Absil, K.A. Gallivan, and P. Hand. ROPTLIB: An object-oriented C++ library for optimization on Riemannian manifolds. Technical Report FSU16-14, Florida State University, 2016.

[27] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J.J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping using the Bayes tree. *Intl. J. of Robotics Research*, 31(2):216–235, February 2012.

[28] K. Khosoussi, S. Huang, and G. Dissanayake. Novel insights into the impact of graph structure on SLAM. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 2707–2714, Chicago, IL, September 2014.

[29] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 3607–3613, Shanghai, China, May 2011.

[30] C.-J. Lin and J.J. Moré. Incomplete Cholesky factorizations with limited memory. *SIAM J. Sci. Comput.*, 21(1): 24–45, 1999.

[31] J. Mandel. On block diagonal and Schur complement preconditioning. *Numerische Mathematik*, 58(1):79–93, December 1990.

[32] D. Martinec and T. Pajdla. Robust rotation and translation estimation in multiview reconstruction. In *IEEE Intl. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, Minneapolis, MN, June 2007.

[33] J.L. Morales and J. Nocedal. Automatic preconditioning by limited memory quasi-Newton updating. *SIAM J. Optim.*, 10(4):1079–1096, 2000.

[34] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer Science+Business Media, New York, 2nd edition, 2006.

[35] E. Olson, J.J. Leonard, and S. Teller. Fast iterative alignment of pose graphs with poor initial estimates. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 2262–2269, Orlando, FL, May 2006.

[36] D.M. Rosen, M. Kaess, and J.J. Leonard. RISE: An incremental trust-region method for robust online sparse least-squares estimation. *IEEE Trans. on Robotics*, 30 (5):1091–1108, October 2014.

[37] D.M. Rosen, C. DuHadway, and J.J. Leonard. A convex relaxation for approximate global optimization in simultaneous localization and mapping. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 5822–5829, Seattle, WA, May 2015.

[38] D.M. Rosen, L. Carlone, A.S. Bandeira, and J.J. Leonard. A certifiably correct algorithm for synchronization over the special Euclidean group. In *Intl. Workshop on the Algorithmic Foundations of Robotics (WAFR)*, San Francisco, CA, December 2016.

[39] D.M. Rosen, L. Carlone, A.S. Bandeira, and J.J. Leonard. SE-Sync: A certifiably correct algorithm for synchronization over the special Euclidean group. Technical Report MIT-CSAIL-TR-2017-002, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA, February 2017.

[40] Y. Saad. On the rates of convergence of the Lanczos and the block-Lanczos methods. *SIAM J. Numer. Anal.*, 17 (5):687–706, October 1980.

[41] A. Singer and H.-T. Wu. Vector diffusion maps and the connection Laplacian. *Comm. Pure Appl. Math.*, 65: 1067–1144, 2012.

[42] C. Stachniss, J.J. Leonard, and S. Thrun. Simultaneous localization and mapping. In *Springer Handbook of Robotics*, pages 1153–1176. Springer International Publishing, 2016.

[43] T. Steihaug. The conjugate gradient method and trust regions in large scale optimization. *SIAM J. Numer. Anal.*, 20(3):626–637, June 1983.

[44] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT Press, Cambridge, MA, 2008.

[45] M.J. Todd. Semidefinite optimization. *Acta Numerica*, 10:515–560, 2001.

[46] R. Tron, B. Afsari, and R. Vidal. Intrinsic consensus on SO(3) with almost-global convergence. In *IEEE Conf. on Decision and Control*, pages 2052–2058, Maui, HI, December 2012.

[47] H. Wang, G. Hu, S. Huang, and G. Dissanayake. On the structure of nonlinearities in pose graph SLAM. In *Robotics: Science and Systems (RSS)*, Sydney, Australia, July 2012.