# Verification Techniques for Pose Graph Optimization

pweinger@stanford.edu

***Abstract*—This paper presents a comprehensive analysis of verification techniques for Pose Graph Optimization (PGO). Leveraging the foundational SE-Sync algorithm, we develop custom algorithm implementations and tests for an in-depth evaluation. The PGO problem is transformed into a relaxed Semidefinite Programming (SDP) problem to obtain a quality certificate. From the relaxed SDP solution, we derive a feasible PGO solution. In many practical cases, the relaxed and original problems share the same optimal value, resulting in a quality certificate with tight bounds. However, understanding the limitations of SDP relaxations is crucial. We design custom tests with non-tight bounds to assess the performance of different algorithms. Our main contribution is the exploration of an alternative approach based on Riemannian optimization over a product of SO$(3)$ manifolds. When certificate bounds are not tight, this method generally produces superior PGO solutions.**

## I. Problem Formulation

## II. Introduction

We consider the problem of Pose Graph Optimization (PGO), which consists of estimating a set of poses, rotations, and translations from pairwise relative pose measurements. This problem is typically formulated as a maximum a posteriori (MAP) or maximum-likelihood estimation (MLE) under an assumed probability distribution for the measurement noise, which results in a nonlinear least squares problem. Standard approaches to solving PGO employ iterative methods such as Gauss-Newton or Levenberg-Marquardt, which achieve locally optimal solutions. However, these methods do not guarantee the quality of the solution, as convergence to different local optima can vary depending on the initial conditions. Ideally, we seek to understand how close we are to the global optimum and whether further solution refinement is necessary, especially in safety-critical applications.

In the past decade, duality-based verification techniques for Simultaneous Localization and Mapping (SLAM) have gained significant attention. A notable advancement in this field is SE-Sync [1]: a certifiably correct algorithm for synchronization over the Special Euclidean Group. SE-Sync leverages Lagrangian duality [2] and Semidefinite Programming (SDP) [3] relaxations to provide optimality bounds. This paper reviews the prior work that led to the development of SE-Sync, analyzes the SE-Sync algorithm itself, and discusses some of the remaining challenges in 2024 related to duality-based verification techniques for PGO.

## III. Literature Review

In [4], Carlone and Dellaert present a technique to verify if a given 2D SLAM solution is globally optimal. They establish lower and upper bounds for the optimal solution, enabling the assessment of the quality of any solution returned by an iterative solver. By re-parameterizing the rotation matrices $R_i \in \mathrm{SO}(2)$, they reformulate the initial nonlinear least-squares minimization problem into a Quadratically Constrained Quadratic Programming (QCQP) minimization problem. They reduce the dimensionality of the problem by eliminating the position vectors using the Schur complement. Subsequently, they compute the dual problem of the QCQP, providing a method to verify global optimality.

Lagrangian duality theory provides two key insights: the dual problem is always convex, regardless of the primal problem's nature, and the optimal value of the dual problem offers a lower bound on the primal's optimal value. In cases where strong duality holds, this bound is tight. The dual problem is an SDP (Semidefinite Programming) problem, solvable by off-the-shell convex solvers. Any feasible solution to the primal problem provides an upper bound for the minimization problem, ensuring that the solution quality can be effectively assessed. Experimental results indicate that, in all tested cases, the bound provided by the dual problem was tight, suggesting that strong duality may hold, even if not formally proven.

However, a significant challenge remains. Pose Graph Optimization (PGO) is a large-scale problem requiring the estimation of thousands of poses, making the dual SDP problem difficult for current off-the-shelf solvers to handle efficiently. With most real datasets, solving the SDP problem required approximately one hour. Despite the appealing quality of the certificate gap, the framework is impractical for real-world application due to the scalability issues of off-the-shelf SDP solvers.

In a follow-up paper [5], the same authors generalize their approach to 3D SLAM. In this work, while re-parameterizing the matrices in $\mathrm{SO}(3)$, they drop the constraint $\det(R) = 1$, leading to a QCQP minimization problem where the equality constraints correspond to matrices being part of $\mathrm{O}(3)$ instead of $\mathrm{SO}(3)$. Consequently, the resulting bounds are looser. However, given a candidate solution we can still check if a solution is optimal and still provides a certificate gap, albeit a looser one. Additionally, they introduce a technique to verify whether a candidate solution is optimal without solving a large-scale SDP problem. This method relies on solving a linear system and checking the positive semi-definite nature of a matrix. This test is fast, but the result is binary, qualifying a solution as optimal or not, without providing bounds. Experimental results demonstrate that the bounds are tight when the noise level, modeled as a Gaussian isotropic noise $\mathcal{N}\left(0,\, \sigma_R^2 I\right)$, is below a certain threshold $\sigma_R = 0.1$.

However, the bounding gap increases as $\sigma_R$ increases. The certificate quality decreases with increasing noise levels.

In [6], Briales and Gonzalez-Jimenez improve upon the results from [5] by providing a novel formulation of the QCQP problem that results in smaller matrices. The previous work systematically uses the Kronecker product to transform matrix products into vector products, resulting in each rotation matrix producing a $9\times 9$ block. In contrast, this paper employs a trace-based reformulation, keeping the rotation matrices as $3 \times 3$ blocks. Additionally, some diagonal blocks corresponding to constant terms are dropped, leading to sparser matrices. This reformulation preserves the quality of the results while reducing computation time by a factor of 50. However, solving the SDP problem with more than 1000 poses remains prohibitively slow, requiring more than 15 minutes.

In [7], Briales and Gonzalez-Jimenez, and in [8], Brynte et al. consider the applicability of these techniques to different problems, including the SLAM front-end. They use these techniques for registration problems (e.g., point cloud registration) with point-to-point, point-to-line, and point-to-plane correspondences. They also apply them to hand-eye calibration and rotation averaging, to name a few. The fewer poses to estimate, the smaller the SDP problem, and the more applicable these techniques are.

In [7], the authors maintain the constraint $\det(R) = 1$ when deriving the QCQP formulation. They replace the default cubic determinant constraint with quadratic constraints in the form of three cross products $R^{(i)} \times R^{(j)} = R^{(k)}$ for $i, j, k = \text{cyclic}\,(1, 2, 3)$. This ensures that each column of a rotation matrix adheres to the right-hand rule, resulting in a rotation matrix instead of a reflection matrix. Consequently, we have 9 additional constraints per rotation matrix. Adding these constraints remarkably improves the quality of the duality gap. In [7], the authors achieve tight experimental results regarding the duality gap. This approach particularly applies to smaller-scale SDP problems, such as those related to SLAM front-end processing.

In [9], Rosen and Carlone design a custom SDP solver to enable fast, real-time computations on large-scale PGO problem sets. Solving the SDP problem requires finding an SDP matrix $Z = VV^T$ of dimension $dn\times dn$, where $d = 3$ and $n$ corresponds to a few thousand poses. However, in practice, the solution $Z$ is of rank $r$, not much greater than $d$. Thus, the main idea is to search for $Z = V_{nd\times r}V^T_{r\times nd}$. This approach dramatically reduces the search space size and renders the positive semidefiniteness constraint redundant since $VV^T \succeq 0$ for any choice of $V$. Consequently, the rank-restricted form of the problem becomes a low-dimensional nonlinear program instead of a semidefinite program. In [10], Burer and Monteiro originally proposed a method to solve this problem based on an augmented Lagrangian procedure. However, Rosen, Carlone et al. in [1], [9] adopt Riemannian optimization. The problem we have to solve is of the form

$$\begin{aligned} \min_{V\in\mathbb{R}^{dn\times r}} \quad & \operatorname{Tr}\left(CVV^T\right) \\ s.t. \quad & \operatorname{Tr}\left(A_iVV^T\right) = b_i \quad i = 1, \dots, m \end{aligned}$$

The set

$$\mathcal{M} = \left\{V \in \mathbb{R}^{dn\times r} \mid \operatorname{Tr}\left(A_iVV^T\right) = b_i, i = 1, \dots, m\right\}$$

is a smooth Riemannian manifold under certain conditions on $A_i$, as explained by Majumdar et al. in [11]. The objective function $V \mapsto \operatorname{Tr}\left(CVV^T\right)$ is smooth, making this problem a Riemannian optimization problem. If $m < \frac{r(r+1)}{2}$, any second-order critical point is globally optimal and Riemannian trust-region methods can return such a point. These principles underpin the real-time SDP solver presented in [9]. In their experiments, the SE-Sync solution [1], regularly enhanced by their research findings, outperformed GTSAM regarding speed while providing an optimality certificate. This research culminated in the seminal paper on SE-Sync [1], which has significantly influenced the field. We have reviewed the origin and evolution of the key concepts described in [1].

In [12], Holmes, Dümbgen and Barfoot emphasize that certifiable methods like SE-Sync rely on simplifying assumptions to facilitate problem formulation, notably assuming an isotropic measurement noise distribution. They address a localization problem, specifically estimating a sequence of poses based on measurements from known landmarks using stereo-camera data. The conversion of stereo pixels to Euclidean coordinates results in a noise distribution that should not be modeled isotropically. Consequently, the resulting maximum-likelihood optimization incorporates matrix rather than scalar weighting factors. Their experiments reveal that semidefinite relaxations were tight only at lower noise levels when matrix weighting factors were used instead of scalar ones.

## IV. Proposed Work

We developed a custom SE-Sync implementation in Python from scratch, detailing the steps involved in transforming the original Pose Graph Optimization (PGO) problem into the final relaxed SDP problem. Our primary objective is to revisit Appendix B of the SE-Sync paper [1], reformulating the estimation problem from problem 1 to problem 7 in a didactic manner to enhance comprehensibility for a broader audience. To solve the SDP problem, we utilize pymanopt, a Python toolbox for optimization on Riemannian manifolds. For comparative benchmark results with various SDP solvers (MOSEK, SDPLR, SDPNAL+, STRIDE, ManiSDP), we refer readers to to Wang et al. [13].

Understanding the limitations of SDP relaxations is crucial. Therefore, we develop custom tests without tight certificate bounds to evaluate the performance of various algorithms. As our main contribution, we investigate an alternative method based on Riemannian optimization over a product of SO(3) manifolds.

## V. Problem Formulation

We consider the following PGO optimization problem:

$$f^*_{\text{ML}} = \min_{t_i,R_i} \sum_{(i,j)\in\mathcal{E}} \omega_t^2 \left\|t_j - t_i - R_i\tilde{t}_{ij}\right\|_2^2 + \frac{\omega_R^2}{2} \left\|R_j - R_i\tilde{R}_{ij}\right\|_F^2$$

with $t_i \in \mathbb{R}^3, R_i \in \text{SO}(3)\,, i$, and $j$ vertices, and $\mathcal{E}$ a set of edges connecting vertices. We note $n$ the number of vertices.

Via the Kronecker product, we vectorize the rotation matrices to vectors $r_i \in \mathbb{R}^9$:

$$\text{vec}\left(I_3 R_i \tilde{t}_{ij}\right) = \left(\tilde{t}_{ij}^T \otimes I_3\right) \text{vec}\left(R_i\right) = T_{ij} r_i$$

$$\text{vec}\left(I_3 R_i \tilde{R}_{ij}\right) = \left(\tilde{R}_{ij}^T \otimes I_3\right) \text{vec}\left(R_i\right) = Q_{ij} r_i$$

Leading to a vectorized form:

$$f_{\text{ML}}^* = \min_{t_i, r_i} \sum_{(i,j)\in\mathcal{E}} \omega_t^2 \left\|t_j - t_i - T_{ij} r_i\right\|_2^2 + \frac{\omega_R^2}{2} \left\|r_j - Q_{ij} r_i\right\|_2^2$$

Let $x \in \mathbb{R}^{12n}$ denote the vector obtained by vertically stacking all $t_i \in \mathbb{R}^3$ and $r_i \in \mathbb{R}^9$ vertically for $i \in [1, n]$. We construct extractors $B_{ij} \in \mathbb{R}^{3\times 12n}$ and $C_{ij} \in \mathbb{R}^{9\times 12n}$ such that:

$$\begin{cases} B_{ij} x = t_j - t_i - T_{ij} r_i \\ C_{ij} x = r_j - Q_{ij} r_i \end{cases}$$

With:

$$\begin{cases} B_{ij} = \begin{bmatrix} T_{3\times3}^{ij,1} & \dots & T_{3\times3}^{ij,n} & Q_{3\times9}^{ij,1} & \dots & Q_{3\times9}^{ij,n} \end{bmatrix} \\ T_{3\times3}^{ij,j} = I_3, T_{3\times3}^{ij,i} = -I_3, Q_{3\times9}^{ij,1} = -T_{ij}, \text{other blocks} = 0 \end{cases}$$

$$\begin{cases} C_{ij} = \begin{bmatrix} 0_{9\times3}^{ij,1} & \dots & O_{9\times3}^{ij,n} & Q_{9\times9}^{ij,1} & \dots & Q_{9\times9}^{ij,n} \end{bmatrix} \\ Q_{9\times9}^{ij,j} = I_9, Q_{9\times9}^{ij,i} = -Q_{ij}, Q_{9\times9}^{ij,k} = 0 \text{ for } k \notin \{i,j\} \end{cases}$$

We reformulate the problem as:

$$\begin{cases} f_{\text{ML}}^* = \min_x \quad x^T Q x \\ \text{with } x \in \mathbb{R}^{12n}, Q \in \mathbb{R}^{12n\times 12n} \\ \text{and Q=} \sum_{(i,j)\in\mathcal{E}} \omega_t^2 B_{ij}^T B_{ij} + \frac{\omega_R^2}{2} C_{ij}^T C_{ij} \end{cases}$$

As $n$ typically ranges from a few hundred to several thousand, we aim to reduce the problem's dimensionality as much as possible. To begin, we will eliminate the translation component using the classical Schur complement technique:

$$f(x) = f(t, r) = \begin{bmatrix} t & r \end{bmatrix}^T \begin{bmatrix} Q_{tt} & Q_{tr} \\ Q_{rt} & Q_{rr} \end{bmatrix} \begin{bmatrix} t \\ r \end{bmatrix}$$

$$f(t, r) = t^T Q_{tt} t + 2 r^T Q_{rt} t + r^T A_{rr} r$$

An optimality condition is:

$$\frac{\partial f}{\partial t}(t, r) = 2 Q_{tt} t + 2 A_{rt}^T r = 0$$

Leading to:

$$t^* = -Q_{tt}^{-1} Q_{tr} r$$

By substituting back into $f$ we get:

$$f(x) = f(r) = r^T \left(Q_{rr} - Q_{rt} Q_{tt}^{-1} Q_{tr}\right) r$$

Resulting in a lower dimension problem:

$$\begin{cases} f_{\text{ML}}^* = \min_r \quad r^T Q_r r \\ \text{with } r \in \mathbb{R}^{9n}, Q_r \in \mathbb{R}^{9n\times 9n} \\ \text{and } Q_r = Q_{rr} - Q_{rt} Q_{tt}^{-1} Q_{tr} \end{cases}$$

We further reduce the problem dimensionality by noticing that $Q_r \in \mathbb{R}^{9n\times 9n}$ can be reduced to $\tilde{Q} \in \mathbb{R}^{3n\times 3n}$ . As we started with:

$$Q_{ij} r_i = \left(\tilde{R}_{ij}^T \otimes I_3\right) \text{vec}\left(R_i\right)$$

We actually have:

$$Q_r = \tilde{Q} \otimes I_3$$

We also prove that:

$$r^T Q_r r = \text{Trace}\left(\tilde{Q} R^T R\right)$$

With $R_i \in \text{SO}(3), R = \begin{bmatrix} R_1 & \dots & R_n \end{bmatrix} \in \mathbb{R}^{3\times 3n}, \tilde{Q} \in \mathbb{R}^{3n\times 3n}$, we have $\text{vec}(R) = r$. From elementary properties:

$$\begin{cases} \text{Trace}\left(A^T B\right) = \text{vec}(A)^T \text{vec}(B) \\ \text{vec}(AXB) = \left(B^T \otimes A\right) \text{vec}(X) \end{cases}$$

We derive:

$$\text{Trace}\left(R^T (AXB)\right) = \text{vec}(R)^T \left(B^T \otimes A\right) \text{vec}(X)$$

With $A = I_3, X = R_{3\times 3n}, B = \tilde{Q}^T = \tilde{Q}_{3n\times 3n}$, we obtain:

$$\text{Trace}\left(R^T R \tilde{Q}\right) = \text{vec}(R)^T \left(\tilde{Q} \otimes I_3\right) \text{vec}(R)$$

$$\text{Trace}\left(\tilde{Q} R^T R\right) = r^T Q_r r$$

Leading to our final problem formulation:

$$\begin{cases} f_{\text{ML}}^* = \min_R \quad \text{Trace}\left(\tilde{Q} R^T R\right) \\ \text{with } R \in \text{SO}(3)^n, \tilde{Q} \in \text{Sym}(3n) \end{cases}$$

By taking $X = R^T R \succeq 0$, this problem can be reformulated as a Semidefinite Programming problem. We relax the constraint $R \in \text{SO}(3)^n$ to $R \in \text{O}(3)^n$ leading to $X$ with identity diagonal blocks corresponding to $R_i^T R_i = I_3$ We also know that $R$ is of rank 3, thus $X = R^T R$ is of rank 3. But in the below SDP formulation we ignore the constraint $\text{rank}(X) = 3$ to get a convex problem.

$$\begin{cases} f_{\text{SDP}}^* = \min_{X\in\text{Sym}(3n)} \quad \text{Trace}\left(\tilde{Q} X\right) \\ \text{s.t. } X = \begin{bmatrix} I_3 & * & * \\ * & \ddots & * \\ * & * & I_3 \end{bmatrix} \succeq 0 \end{cases}$$

This is indeed a convex SDP problem. The objective is linear in $X$. The set of PSD matrices is convex. The set of matrices that are positive semidefinite and have $I_3$ diagonal blocks is convex. If we consider two matrices $X_1, X_2$ PSD with $I_3$ diagonal blocks then $\lambda X_1 + (1-\lambda) X_2$ is PSD and still has $I_3$ diagonal blocks. Thus we can find a global optimum to $f_{\text{SDP}}$. By solving these two problems $f^*_{\text{ML}}$ and $f^*_{\text{SDP}}$, we not only get a solution for our initial PGO problem, but we also get a quality certificate. The initial problem $f^*_{\text{ML}}$ is a non convex problem as the constraints are non convex. By solving $f^*_{\text{ML}}$ we are only guaranteed to converge to a local minimum. There may be better solutions, but no other solution can reach a lower bound than the SDP convex relaxation $f^*_{\text{SDP}}$. If $f^*_{\text{ML}} = f^*_{\text{SDP}}$ we are certified to have found a global minimizer for our initial PGO problem. This relaxed SDP problem can be derived in an alternative way, by computing the dual of the dual of the original $f_{\text{ML}}$ problem, as demonstrated in section 4 and B4 of [1].

## VI. Solution methods

We will solve this problem in three different ways, providing additionally a lower bound for the a priori unknown optimal solution.

While PGO problems are typically solved with Non Linear Least Squares methods like e.g. Levenberg-Marquardt method we investigate optimization on Riemannian Manifolds.

### *A. Solving $f_{SDP}$ and $f_{ML}$ with Semidefinite Programming*

To solve $f^*_{\text{SDP}}$, a convex SDP problem, we will first use cvxpy with the SCS solver. This method may struggle to scale to larger problems with $n$ above a few hundred.

After obtaining a solution for $f^*_{\text{SDP}}$ , which is a relaxed version of our PGO problem, we will extract a feasible solution for $f_{\text{ML}}$. The solution to $f^*_{\text{SDP}}$ is a matrix $X$ of size $3n \times 3n$ which we want to factorize as $X = R^T R$ with $R \in \text{SO}(3)^n$ of size $3 \times 3n$ We perform a SVD decomposition of $X = U \Sigma V^T$. Since $R$ is of rank 3, we reconstruct $R$ with the 3 largest singular values and their corresponding singular vectors. Let $U_i$ be the $i^{th}$ column of $U$, then:

$$R = \begin{bmatrix} \sqrt{\sigma_1} V_1^T & \sqrt{\sigma_2} V_2^T & \sqrt{\sigma_3} V_3^T \end{bmatrix}$$

To obtain the nearest rotation matrix, we perform a subsequent SVD decomposition:

$$R = U_r \Sigma_r V_r^T$$

This gives:

$$R_{\text{nearest rotation}} = U_r V_r^T$$

If $\det\left(U_r V_r^T\right) < 0$, we adjust the sign of the last column of $U_r$, to convert the reflection to a rotation.

### *B. Solving $f_{ML}$ with Riemannian optimization over a SO$(3)$ manifold*

We begin by introducing the concept of Riemannian optimization over a manifold. A manifold is a concept in mathematics that generalizes the idea of surfaces to higher dimensions. Formally, a manifold is a topological space that locally resembles a Euclidean space e.g. $\mathbb{R}^n$. Each point on a manifold has a neighborhood that can be mapped to an open subset of $\mathbb{R}^n$ via a continuous, bijective function called a chart. A chart is a pair $(U, \phi)$ where $U$ is an open subset of the manifold and $\phi : U \to \mathbb{R}^n$ is a homeomorphism (a continuous function with a continuous inverse) mapping $U$ to an open subset of $\mathbb{R}^n$. An atlas is a collection of such charts covering the entire manifold. The dimension of a manifold is the dimension of the Euclidean space that it locally resembles. A smooth manifold has an atlas with differentiable transition maps. A Riemannian manifold is a smooth manifold with an inner product on the tangent space at each point. SO$(3)$ is a 3-dimensional smooth manifold. The exponential map relates the tangent space at a point to the manifold itself, moving a tangent vector along a geodesic. Geodesics are the shortest paths between points. On SO$(3)$, they are given by $R(t) = R_0 \exp(t\hat{w})$ where $R_0$ is the initial point, $\hat{w}$ is a constant skew-symmetric matrix and $t$ a parameter.

Riemannian optimization involves optimizing over Riemannian manifolds, reformulating problems as unconstrained problems and leveraging the natural geometry of the problem for potentially faster convergence, compared to general-purpose solvers.

We use pymanopt toolbox to solve $f^*_{\text{ML}}$. By employing a first-order unconstrained optimizer, such as Conjugate Gradient, we aim to handle large-scale problems effectively. Additionally, we will experiment with a second-order Trust Region method. The information we need to provide to the solver includes the cost function $\min\ \text{Trace}\left(\tilde{Q} R^T R\right)$, the gradient function $2\ R\tilde{Q}$, the hessian function $2\ \tilde{Q}$ for the second-order method, and the manifold we want to operate on, which is actually a product of $n$ SO$(3)$ manifolds. Products of manifolds are indeed manifolds.

### *C. Solving $f_{SDP}$ and $f_{ML}$ with Riemannian optimization over a Stiefel manifold*

SDP solvers do not scale for $n \geq 1000$ when dealing with:

$$\begin{cases} f^*_{\text{SDP}} = \min\limits_{X \in \text{Sym}(3n)} \quad \text{Trace}\left(\tilde{Q} X\right) \\ \text{s.t. } X \succeq 0, \text{blocks diagonal} X_{ii} = I_3 \end{cases}$$

We note that we should expect low-rank solutions. As demonstrated by Pataki in [14], the problem has a solution of rank$\leq \sqrt{n(d+1)}$. Additionally, our underlying problem calls for a low-rank solution: we replaced $R^T R$ of rank 3 with $X$. The SDP low-rank idea from Burer et al. [10] involves factorizing $X$ with a tall and skinny matrix $Y$, leading to:

$$\begin{array}{cc} \min\limits_{Y} & \text{Trace}\left(QYY^T\right) \\ s.t. & X = YY^T \\ & Y \in \mathbb{R}^{n \times p} \text{ leading to rank}(X) \leq p \end{array}$$

This approach dramatically reduces the size of the search space and makes the PSD constraint redundant, since $YY^T \succeq 0$. However, the resulting rank-restricted form of the problem is no longer an SDP problem and is no longer convex. Instead, we are dealing with a low-dimensional nonlinear program. In [15], [16], Boumal et al. derived the Riemannian staircase algorithm to solve this problem.

We us the following change of variable:

$$
\begin{array}{lcl}
X = X^T \succeq 0 & & \exists Y \in \mathbb{R}^{n\times p} \text{ such that} X = YY^T \\
\operatorname{rank}(X) \leq p & \text{changed to} & Y = \begin{bmatrix} Y_1 \\ \vdots \\ Y_n \end{bmatrix}, Y_i \in \mathbb{R}^{3\times p} \\
X_{ii} = I_3 \forall i & & Y_i Y_i^T = I_3 \forall i
\end{array}
$$

$Y_i$ is $3 \times p$ orthonormal for $i = 1, \ldots, m$ and belongs to a Stiefel manifold whereas $Y$ belongs to a Stiefel $(3, p)^n$ manifold. Boumal et al. established a connection between these two problems:

$$
\begin{array}{cc}
\text{linear, convex} & \text{non linear, non convex} \\
\min_X f(X) & \min_Y f\left(YY^Y\right) \\
X \succeq 0, X_{ii} = I_d & Y \in \text{Stiefel}\,(d, p)^n
\end{array}
$$

If $Y$ is a local minimizer of the nonlinear program and if $Y$ is rank deficient or if $Y$ is square then $X = YY^T$ is a global minimizer of the initial convex program.

This leads to the Riemannian Staircase Algorithm:

1) Set $p = d + 1$.
2) Compute $Y_p$, a Riemannian local optimizer.
3) If $Y_p$ is rank deficient, stop.
4) Otherwise, increase $p$ and go to step 2.

This is guaranteed to return a globally optimal $X$.

We initially attempted to implement this algorithm using pymanopt. However, we had to switch to the matlab manopt version because the manifold Stiefel $(d, p)^n$ in the python version does not support $p > d$. The Matlab version supports this via a dedicated stiefelstackedfactory manifold. After solving $f_{\text{SDP}}$, we use the same technique presented above, based on SVD decomposition, to retrieve a feasible solution for $f_{\text{ML}}$.

## VII. Experiments

We provide an implementation for the three algorithms described above in SORI and benchmark them against SE-Sync. Our evaluation consists of two batches of tests. The first batch uses the test suite provided by SE-Sync, while the second batch includes a custom set of tests that we developed.

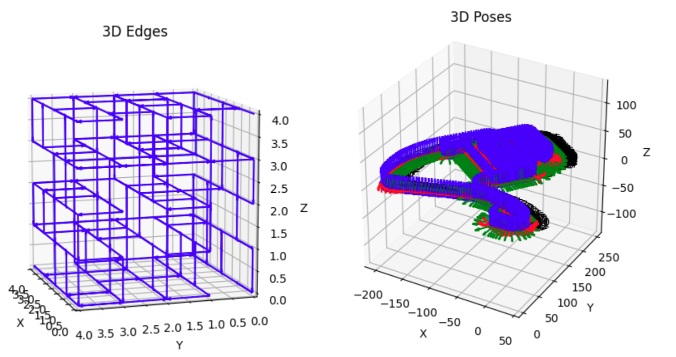

Figure 1: Grid3D N=4 and parking-garage tests

Our custom tests include a random test and a scalable 3D grid path test. In the 3D grid path test, we simulate a robot navigating from one corner of a cube to the opposite corner, sequentially visiting all integer coordinate points within the cube. While the robot's 3D position evolves regularly, its orientation is randomly generated at each time step. Relative measurements are obtained by adding Gaussian noise with $\sigma_T = 0.1$ m for translations and $\sigma_R = 0.05$ rad for rotations. Additionally, loop closures are introduced between nearby points with a probability of 0.3.

In all cases and for all algorithms, we start with a random solution to increase the problem's difficulty and to verify the repeatability of the SDP solutions. The results of the tests are presented below. We evaluate the solutions obtained for the $f_{\text{SDP}}$ and $f_{\text{ML}}$ problems, as well as the run time performance of the different algorithms.

Table I: SE-Sync tests

| Dataset | Sol. | $\hat{f}_{\text{SDP}}$ | $\hat{f}_{\text{ML}}$ | $\left\lvert\frac{\hat{f}_{\text{SDP}}-\hat{f}_{\text{ML}}}{\hat{f}_{\text{SDP}}}\right\rvert$ | Time |
|---|---|---|---|---|---|
| tinyGrid3D | sdp scs | 18.52 | 18.52 | 0 | 0.05 s |
| $n = 9$ | riemann so(3) | - | 18.52 | 0 | 0.08 s |
| $m = 11$ | sdp staircase | 18.52 | 18.52 | 0 | 0.03 s |
| | se-sync | 18.52 | 18.52 | 0 | 0.13 s |
| smallGrid3D | sdp scs | 1025.4 | 1025.4 | 0 | 167 s s |
| $n = 125$ | riemann so(3) | - | 1025.4 | 0 | 3.4 s |
| $m = 297$ | sdp staircase | 1025.4 | 1025.4 | 0 | 0.3 s |
| | se-sync | 1025.4 | 1025.4 | 0 | 0.35 s |
| parking | sdp scs | - | - | - | - |
| $n = 1661$ | riemann so(3) | - | 369 | 282% | 600 s time out |
| $m = 6275$ | sdp staircase | 1.37 | 1.37 | 0 | 188 s |
| | se-sync | 1.26 | 1.26 | 0 | 206 s |

Table II: Custom tests

| Dataset | Sol. | $\hat{f}_{\text{SDP}}$ | $\hat{f}_{\text{ML}}$ | $\left\lvert\frac{\hat{f}_{\text{SDP}}-\hat{f}_{\text{ML}}}{\hat{f}_{\text{SDP}}}\right\rvert$ | Time |
|---|---|---|---|---|---|
| random | sdp scs | 25716 | 27587 | 7.3% | 0.05 s |
| $n = 9$ | riemann so(3) | - | **26947** | **4.8%** | 0.19 s |
| $m = 25$ | sdp staircase | 25716 | 27587 | 7.3% | 0.03 s |
| | se-sync | 25716 | 27587 | 7.3% | 0.03 s |
| Grid3D N=2 | sdp scs | 1549 | 1549 | 0 | 4.6 s |
| $n = 27$ | riemann so(3) | - | 2158 | 39.3% | 0.78 s |
| $m = 30$ | sdp staircase | 1549 | 1549 | 0 | 0.26 s |
| | se-sync | 1549 | 1549 | 0 | 0.29 s |
| Grid3D N=3 | sdp scs | 6760 | 11181 | 65.4% | 66.0 s |
| $n = 64$ | riemann so(3) | - | **10292** | **52.2%** | 2.64 s |
| $m = 81$ | sdp staircase | 6760 | 1549 | 65.4% | 0.6 s |
| | se-sync | 6760 | 11166 | 65.4% | 0.6 s |
| Grid3D N=4 | sdp scs | 65355 | 78523 | 20.1% | 428.0 s |
| $n = 125$ | riemann so(3) | - | **74597** | **14.1%** | 7.15 s |
| $m = 283$ | sdp staircase | 65352 | 78534 | 20.1% | 2.2 s |
| | se-sync | 65352 | 78534 | 20.1% | 2.1 s |
| Grid3D N=5 | sdp scs | 109424 | 136109 | 26.2% | 600 s timeout |
| $n = 216$ | riemann so(3) | - | **131774** | **22.1%** | 22.1 s |
| $m = 482$ | sdp staircase | 107918 | 137259 | 27.1% | 3.2 s |
| | se-sync | 107947 | 137259 | 27.1% | 3.1 s |
| Grid3D N=6 | sdp scs | - | - | - | - |
| $n = 343$ | riemann so(3) | - | **222550** | **22.1%** | 22.1 s |
| $m = 783$ | sdp staircase | 190437 | 254583 | 16.9% | 4.5 s |
| | se-sync | 190400 | 254583 | 33.7% | 4.4 s |
| Grid3D N=7 | sdp scs | - | - | - | - |
| $n = 512$ | riemann so(3) | - | **306603** | **22.1%** | 50.1 s |
| $m = 1140$ | sdp staircase | 260027 | 365511 | 40.5% | 13.0 s |
| | se-sync | 260027 | 365511 | 40.5% | 8.5 s |
| Grid3D N=8 | sdp scs | - | - | - | - |
| $n = 729$ | riemann so(3) | - | **469889** | **16.2%** | 93.2 s |
| $m = 1686$ | sdp staircase | 404610 | 541615 | 33.9% | 24.4 s |
| | se-sync | 404494 | 541590 | 33.9% | 14.2 s |
| Grid3D N=9 | sdp scs | - | - | - | - |
| $n = 1000$ | riemann so(3) | - | **618526** | **18.4%** | 115.3 s |
| $m = 2280$ | sdp staircase | 522279 | 707652 | 35.4% | 26.7 s |
| | se-sync | 522548 | 707672 | 35.4% | 14.2 s |

As expected, the standard SDP solvers do not scale well for the PGO problem. To address this, we employ the SDP low-rank staircase algorithm, which is highly efficient and allows us to derive a feasible solution to the $f_{\text{ML}}$ problem from an $f_{\text{SDP}}$ solution. For the SE-sync batch of tests, this approach proves to be the preferred solution, as the bounds are consistently tight, yielding $f^*_{\text{SDP}} = f^*_{\text{ML}}$.

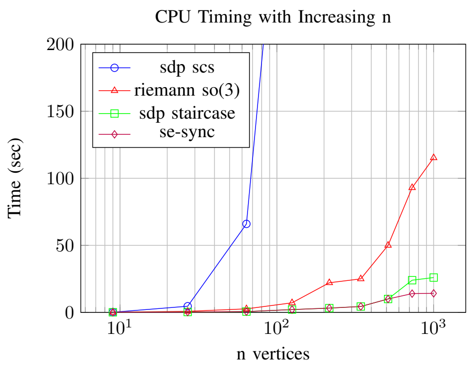

Figure 2: CPU Timing with Increasing n

However, in our custom batch of tests, the situation is different. In almost all cases, the bounds are not tight. In these instances, the second algorithm, which relies on Riemannian optimization over a product of $\text{SO}(3)$ manifolds, provides much better feasible solutions. This is an interesting result, as this technique was not investigated in the literature we reviewed. Nonetheless, this algorithm does not scale as well as the SDP staircase algorithm, even though both are based on Riemannian optimization.

In robotics state estimation, the state of a robot typically includes not just a vector but also rotations or poses. In this context, Riemannian optimization is appealing as it can directly operate on $\text{SO}(d)$, $\text{SE}(d)$ or Stiefel manifolds. For Riemannian $\text{SO}(3)$ optimization, we use a Conjugate Gradient (CG) algorithm with minimal customization effort.

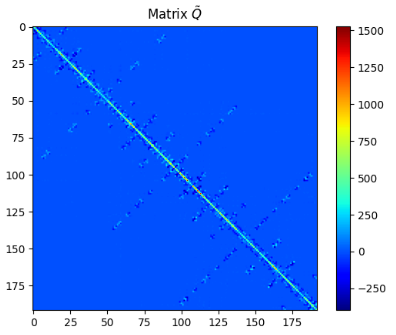

Figure 3: Matrix Sparsity for Grid3D N=3

As highlighted by the above figure, the $\tilde{Q}$ matrix is very sparse, with values concentrated along the diagonal. A preconditioner should be investigated to further enhance the scalability of this algorithm [17].

## VIII. Conclusion

In this study, we performed a comprehensive analysis of verification techniques for Pose Graph Optimization. We developed custom algorithm implementations and test sets to facilitate an detailed evaluation. Our benchmarking of various methods revealed that while standard Semidefinite Programming (SDP) solvers are robust, they do not scale efficiently to larger problems. In contrast, the SDP staircase algorithm, leveraging Riemannian optimization over Stiefel manifolds, offers an efficient approach for obtaining both a quality certificate and a feasible solution. Nevertheless, in cases where the quality certificate bounds are not tight, the method based on Riemannian optimization over a product of SO(3) manifolds consistently produces superior feasible solutions. Future work will focus on deriving a preconditioner to enhance the convergence speed of this method.

## References

[1] David M Rosen, Luca Carlone, Afonso S Bandeira, and John J Leonard. Se-sync: A certifiably correct algorithm for synchronization over the special euclidean group. *The International Journal of Robotics Research*, 38(2-3):95–125, 2019.

[2] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, USA, 2004.

[3] Christoph Helmberg. *Semidefinite programming for combinatorial optimization*. PhD thesis, 2000.

[4] Luca Carlone and Frank Dellaert. Duality-based verification techniques for 2d slam. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 4589–4596. IEEE, 2015.

[5] Luca Carlone, David M Rosen, Giuseppe Calafiore, John J Leonard, and Frank Dellaert. Lagrangian duality in 3d slam: Verification techniques and optimal solutions. in 2015 ieee. In *RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 125–132.

[6] Jesus Briales and Javier Gonzalez-Jimenez. Fast global optimality verification in 3d slam. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4630–4636. IEEE, 2016.

[7] Jesus Briales and Javier Gonzalez-Jimenez. Convex global 3d registration with lagrangian duality. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4960–4969, 2017.

[8] Lucas Brynte, Viktor Larsson, José Pedro Iglesias, Carl Olsson, and Fredrik Kahl. On the tightness of semidefinite relaxations for rotation estimation. *Journal of Mathematical Imaging and Vision*, pages 1–11, 2022.

[9] D Rosen and Luca Carlone. Computational enhancements for certifiably correct slam. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[10] Samuel Burer and Renato DC Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical programming*, 95(2):329–357, 2003.

[11] Anirudha Majumdar, Georgina Hall, and Amir Ali Ahmadi. Recent scalability improvements for semidefinite programming with applications in machine learning, control, and robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:331–360, 2020.

[12] Connor Holmes, Frederike Dümbgen, and Timothy D Barfoot. On semidefinite relaxations for matrix-weighted state-estimation problems in robotics. *arXiv preprint arXiv:2308.07275*, 2023.

[13] Jie Wang and Liangbing Hu. Solving low-rank semidefinite programs via manifold optimization. *arXiv preprint arXiv:2303.01722*, 2023.

[14] Gábor Pataki. On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues. *Mathematics of operations research*, 23(2):339–358, 1998.

[15] Nicolas Boumal. A riemannian low-rank method for optimization over semidefinite matrices with block-diagonal constraints. *arXiv preprint arXiv:1506.00575*, 2015.

[16] Nicolas Boumal, Vlad Voroninski, and Afonso Bandeira. The non-convex burer-monteiro approach works on smooth semidefinite programs. *Advances in Neural Information Processing Systems*, 29, 2016.

[17] C Vuik. Krylov subspace solvers and preconditioners. *ESAIM: Proceedings and Surveys*, 63:1–43, 2018.