Chordal Based Error Function for 3-D Pose-Graph Optimization

This paper has been published on the IEEE Robotics and Automation Letters (RA-L).
Please cite this work as:


```
@article{aloise2019chordal,
  title={Chordal Based Error Function for 3-D Pose-Graph Optimization},
  author={Aloise, Irvin and Grisetti, Giorgio},
  journal={IEEE Robotics and Automation Letters},
  volume={5},
  number={1},
  pages={274--281},
  year={2019},
  publisher={IEEE}
}
```

Chordal Based Error Function for 3D Pose-Graph Optimization

Irvin Aloise Giorgio Grisetti

***Abstract*—Pose-Graph Optimization (PGO) is a well-known problem in the Robotics community. Optimizing a graph means finding the configuration of the nodes that best satisfies the edges. This is generally achieved using iterative approaches that refine a current solution until convergence. Nowadays, Iterative Least-Squares (ILS) algorithms such as Gauss-Newton (GN) or Levenberg-Marquardt (LM) are dominant. Common to all these implementations is the influence of the error function used to measure the difference between prediction and observation. The smoother the error function is, the better the convergence properties of the system become, resulting in an increased convergence basin and more stable behavior. In this work, we propose an alternative error function based on a variant of the Frobenious norm between transformation matrices. The proposed approach leads to a larger convergence basin and to numerical properties in the Jacobian computation that can potentially speed-up the system. In contrast with some existing approximations, our formulation allows to model isotropic and anistropic noise covariances. To validate our conjectures, we present an extensive comparative analysis between our approach and one of the most used error function that computes the distance in the unit-quaternion space.**

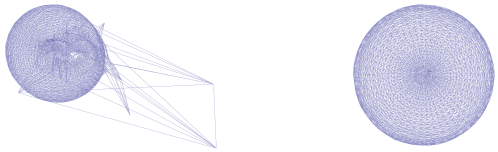
I. Introduction

Mobile robots that have to accomplish non trivial tasks in an unknown environment require a map that describes the surrounding, together with a reasonable estimate of their pose in this representation.

The problem of building the map while estimating the robot pose is known as Simultaneous Localization and Mapping (SLAM) and it has been studied since many years by the community [1] [2]. Many paradigms have been investigated to solve this problem, still, the most promising - and most used nowadays - is the so called *graph-based* approach. Graph-based SLAM systems are known to be efficient and accurate, allowing to have the flexibility and scalability that one expects from current state-of-the-art implementations. As discussed in [3], in this paradigm the SLAM system is built by two main components: a front-end and a back-end. The former is in charge of processing raw data coming from on-board sensors and construct an abstract graph - a stochastic representation of a map. Nodes embody robot poses or the position of generic landmarks - *e.g.* salient world points - while edges express constraints between subsets of nodes, which are computed from the raw data.

Generally, this graph is build incrementally while the robot travels in the environment. Inconsistencies might arise due to sensor aliasing that cause failures in data association, ultimately leading to a corrupted map. The most intuitive way to prevent wrong associations is to restrict the search for potential matches in the map based on the global configuration of the graph. To implement this strategy, the graph should be optimized each time the addition of a new constraint renders the current solution far from the optimum. The back-end is the module in charge of providing such an optimal nodes' configuration, based on the constraints. The front-end, in turn uses this optimal configuration to perform informed searches. It results that desired properties for a back-end are speed, accuracy and having a large convergence basin.

All authors are with the Department of Computer, Control, and Management Engineering "Antonio Ruberti", Sapienza University of Rome, Rome, Italy, Email: {ialoise, grisetti}@diag.uniroma1.it



(a) Standard error function output. (b) Chordal error function output.

Fig. 1: In the first row: output of 100 iteration of Gauss-Newton (GN) optimization using the spanning tree for the initialization, using respectively the standard (1a) and the proposed error function (1b). The AWGN added to the edges has covariance $\Sigma_t = \text{diag}(0.5; 0.5; 0.01)$ and $\Sigma_R = \text{diag}(0.0001; 0.0001; 0.1)$ respectively for the translation and rotation component of the edge.

In the graph-based SLAM paradigm, a pose-graph is a specialization of a generic graph, in which nodes and edges represent robot poses and pose constraints between two nodes, respectively. The problem represents the backbone of SLAM. Many systems [4] [5] [6] are able to perform Pose-Graph Optimization (PGO) in real-time, even on graph with thousands of nodes and edges. The majority of them is based on Iterative Least-Squares (ILS) algorithms, solving a quadratic approximation of the original problem. The ability to solve a problem by an ILS back-end depends mainly on how close the initial guess is to the optimal solution and well a quadratic approximation captures the original non-linear problem. Many works investigated how to compute good initial guess for pose-graphs, using different approaches, an excellent overview is provided by Carlone *et al.* [7].

PGO is a highly non-linear problem, since both nodes and edges involve orientations. The non-linearity of a function might be evaluated by analyzing its derivatives. To the limit, if the error functions have a constant Jacobian matrices, ILS would converge in just one iteration. Clearly this is not the case, since the problem is defined to be non-linear. However by choosing error functions whose Jacobian change mildly leads to a more stable behavior of ILS.

In this work, we propose an error function for 3D PGO that shows a smoother behavior. The proposed approach is based on a variant of the Frobenious norm that and can be seamlessly embedded in existing SLAM pipelines. Our function can be also derived (and thus implemented) rather easily. The increased smoothness, results in an enlarged convergence basin that allows ILS to retrieve the optimal solution in conditions when traditional approaches fail, as illustrated in Fig. 1.

Our claims are supported by a broad range of comparative experiments. We also release an open-source C++ plugin[1] for the well-known optimization framework g^2o that embeds the proposed error function.

II. Related Work

SLAM is a well known problem and it has been studied intensively by the research community in the past years. Many paradigms have been proposed to efficiently solve this problem. Early approaches were based on filtering, proposing solution based on Gaussian Filters (GF) [8] [9] [10] [11] [12] [13] or Particle Filters (PF) [14] [15] [16]. Nonetheless, GF suffer from the intrinsic non-linearity of the problem - attenuated using the Unscented Transform [17]. Furthermore, those systems require to store the entire covariance matrix of the system. This can be prohibitive when the number of variables is large. In contrast, PF can better handle non-linear sensor models and non-Gaussian noise, however, the number of particles involved easily becomes non-treatable in real-time for very large environments. Thanks to increased computational power available on newer machines, Maximum-A-Posteriori (MAP) based approaches started to gain attention, due to their efficiency and accuracy. As a result, the *graph-based* model became the gold standard in last decade to address this kind of problem.

The work of Lu *et al.* [18] represents the first use of this paradigm in SLAM, applying it in the context of laser scans registration. They build a graph whose nodes are either robot poses or points relative to the laser scan acquired from that pose. Edges represent spatial constraints between nodes, obtained both from odometry measurements or by registering pairs of nearby scans. In this scenario, they optimize the graph using a the well-known GN optimization algorithm. Therefore, the final map is obtained through iterative minimization of an objective function that considers the difference between the relative pose between two nodes and the relative pose reported by the scan-matcher. Gutmann and Konolidge [19] addressed the problem of incrementally build a map, finding topological relations and loop closures. To achieve good performances the optimization considers only inconsistent portions of the graph, affecting the accuracy of the solution.

These works were among the first implementations of a full graph-based SLAM pipeline. In these systems, the back-end was seen as a computational bottleneck. Accordingly, the community invested increasing effort in providing more efficient back-ends. Duckett *et al.* [20] proposed an approach leveraging on *Gauss-Seidel relaxation*. In this work they update one node at the time by moving along the gradient of the error function and assuming the neighbors are fixed. Relaxation is easy to implement and memory efficient, since it did not require to store any system matrix. Still, its convergence is slow compared to ILS approaches. To overcome this issue Frese *et al.* [21] proposed a multilevel version of the optimization algorithm. Howard *et al.* [22] extended the use of relaxation to other Robotics tasks, like the calibration of a distributed sensor network.

Olson *et al.* [23] proposed an approach based on Stochastic Gradient Descent (SGD). His method uses represents the node's position as the composition of the increments along a 2D trajectory. By using these representation, updating a constraint can be done in logarithmic time. SGD has shown to be robust to poor initial guesses, and to get close to the optimum. Subsequently, Grisetti *et al.* [24] extended this approach to 3D environments, and modified the representation of the node's position by using the increments on a spanning tree rather than along the trajectory. This alternative further improves the convergence speed.

The optimization of a graph in SLAM is by definition a sparse problem. This is due to the fact that edges connect only a small subset of nodes. This is a consequence of the spatial locality of the problem: only nodes that are close in space can be connected by measurements. The number of constraints that can originate from a node depends on the sensor range and the density of the trajectory. Typical front-end implementations, do not augment the graph if the robot did not travel a minimum distance since the past location. Some more advanced front-ends [25], even reuse existing nodes upon revisits. These strategies bound the node density. When the graph is sparse, the approximated Hessian matrix built by a ILS approach is sparse too. In fact its sparsity pattern reflects the adjacency matrix of the graph. Dallaert *et al.* [26] took advantage of this feature releasing a system called $\sqrt{\text{SAM}}$. $\sqrt{\text{SAM}}$ was the first system in the SLAM literature to highlight and exploit the sparse nature of SLAM in the context of ILS approaches by achieving performances that were at the state of the art. Kaess *et al.* upgraded $\sqrt{\text{SAM}}$ to iSAM [27] by adding incremental optimization. Later iSAM2 [28] presented an alternative formulation to represent the problem and implicitly represent and factorize a sparse system. Kümmerle *et al.* [4] embedded in their framework - called g^2o - similar ideas, together with an effective implementation that allowed the system to reach state-of-the-art performances. The framework offers a great modularity - *e.g.* allowing to use different linear solvers or optimization algorithm - and takes advantage of CPU cache and SIMD instructions to perform fast math operations.

Optimization problems are affected by the initial state configuration. Moreover, graph dimension affects the scalability of system, reducing drastically system performances when the number of nodes and edges grows. To cope with those problems, Ni *et al.* [29] and subsequently Grisetti *et al.* [30] applied *divide and conquer* strategies to perform optimization. The former exploits nested dissections to partition the linear system; the latter, instead, builds several connected local subgraph, resulting in a sparser problem.

While designing a front-end for SLAM, the marginal covariances of the node's estimate play an important role in restricting the candidate data association. Computing this estimate, however is a computationally expensive operation. In SLAM++, Ila *et al.* [31] take advantage of the incremental

[1] Source code: https://srrg.gitlab.io/g2o_chordal_plugin.html

aspect of the problem to propose a fast mean and covariance updates for the estimate. They do not re-linearize the entire graph at each optimization epoch, instead they selectively update the factor contributions in the approximated Hessian matrix and recompute the factorization only for some changed regions. This feature, combined with fast block matrix operations and the use of massively parallel computing units - *e.g.* GPUs - allows SLAM++ to achieve state-of-the-art speed performances.

All those approaches suffer from the non-linearities introduced by the rotational component of the system's state and this leads to weak convergence results when the initial guess has a noisy rotational part. Given the complexity of the problem, the research community proposed many approaches to solve a relaxed version of the problem, that still produces a solution closer to the global optimum of the original one. Martinec *et al.* [32] explored chordal relaxation to easily initialize ill-posed pose-graphs. As highlighted by Carlone *et al.* [7], this initialization technique is able to produce good initial guesses despite its intrinsic simplicity. Cucuringu *et al.* [33] and Arrigoni *et al.* [34] proposed a spectral formulation of the problem, respectively in the context of sensor network localization and camera estimation. Rosen *et al.* [35] explored Semi-Definite Programming (SDP) relaxation in PGO, developing an algorithm - called *SE-Sync* - to *synchronize* generic $\mathrm{SE}(d)$ graphs. However, despite the good results achieved, their approach requires an ad-hoc implementation, that solves the problem in multiple steps - *i.e.* first the rotation component of the object is retrieved and the orthonormality is enforced, then the translation is computed fixing the computed rotation. Briales *et al.* [36] in their system called *Cartan-Sync* extended *SE-Sync* focusing on performances and scalability. They use the Frobenius norm in the objective function and jointly optimize rotation and translation through SDP relaxation. However, they share the implementation drawbacks of *SE-Sync* and assume that the noise covariances are isotropic. More recently, Wang *et al.* [37] investigated the influence of novel error function for $\mathrm{SE}(2)$ objects, that computes the error term relative to the orientation part of the odometry through the Frobenius norm. Still, their work is bounded to $\mathrm{SE}(2)$ objects.

We extend the concept of chordal relaxation for $\mathrm{SE}(3)$ objects - *i.e.* in 3D PGO - embedding it in ILS algorithms. Those represent the de-facto standard in SLAM pipelines, being easy to implement while delivering fast and accurate performances in general. The proposed approach, increases the robustness of the system with respect to measurement noise and provides an enlarged convergence basin while keeping the implementation complexity bounded. Even if our approach does not certify that the retrieved solution is the exact solution of the original problem [38] [39], we provided a substantial number of experiments to show that if the convergence is correct, the two minima configurations are almost identical in the majority of the cases.

III. Pose-Graph Optimization

In this section we give a brief overview of the mathematical concepts to perform PGO. A pose-graph is a factor graph whose nodes represent robot (or sensor) poses and edges encode relative transformation between two nodes. Given the pose-graph $\mathcal{G}$, let $\mathbf{X} = \mathbf{X}_{1:N}$ be nodes and let $\{\langle \mathbf{Z}_{ij}, \mathbf{\Omega}_{ij} \rangle\}$ be the set of edges in graph. Each edge $\mathbf{Z}_{ij}$ connects the i-th node to j-th one, while the associated information matrix $\mathbf{\Omega}_{ij}$ describes the edge uncertainty along each dimension. Given an initial configuration of nodes $\bar{\mathbf{X}}$, optimizing the graph means to find the optimal nodes configuration $\mathbf{X}^\star$ that best satisfies the edges. As a result, the PGO problem minimizes the following cost function:

$$\begin{aligned}\mathbf{X}^\star &= \underset{\mathbf{X}}{\operatorname{argmin}} \sum_{i,j} c_{ij}(\mathbf{X}_i, \mathbf{X}_j) \\ &= \underset{\mathbf{X}}{\operatorname{argmin}} \sum_{i,j} ||\mathbf{e}_{ij}(\mathbf{X}_i, \mathbf{X}_j)||^2_{\mathbf{\Omega}_{ij}}. \end{aligned} \tag{1}$$

The error vector $\mathbf{e}_{ij}(\mathbf{X}_i, \mathbf{X}_j)$ computes the mismatch between measurement $\langle \mathbf{Z}_{ij}, \mathbf{\Omega}_{ij} \rangle$ and its prediction $\hat{\mathbf{Z}}_{ij} = \mathbf{h}_{ij}(\mathbf{X}_i, \mathbf{X}_j)$ - which depends from the current nodes configuration. Hence, supposing to have Euclidean measurements - indicated with $\mathbf{z}_{ij}$ - the error between prediction and observation will be:

$$\mathbf{e}_{ij}(\mathbf{X}) = \mathbf{e}_{ij}(\mathbf{X}_i, \mathbf{X}_j) = \hat{\mathbf{z}}_{ij} - \mathbf{z}_{ij} = \mathbf{h}_{ij}(\mathbf{X}_i, \mathbf{X}_j) - \mathbf{z}_{ij}. \tag{2}$$

Still, in PGO measurements lie in $\mathrm{SE}(n)$ which is a smooth manifold. Hence, to compute Eq. (2) we have to define a proper operator $\boxminus$ that returns an Euclidean vector $\hat{\mathbf{Z}}_{ij} \boxminus \mathbf{Z}_{ij} = \mathbf{\Delta z}_{ij}$ representing the manifold distance mapped onto $\mathbb{R}^n$. Accordingly, if $\mathbf{Z}_{ij} = \hat{\mathbf{Z}}_{ij}$ then the difference vector $\mathbf{\Delta z}_{ij}$ will be zero. As a result, the error vector considering manifold measurements $\langle \mathbf{Z}_{ij}, \mathbf{\Omega}_{ij} \rangle$ will be computed as:

$$\mathbf{e}_{ij}(\mathbf{X}_i, \mathbf{X}_j) = \hat{\mathbf{Z}}_{ij} \boxminus \mathbf{Z}_{ij} = \mathbf{h}_{ij}(\mathbf{X}_i, \mathbf{X}_j) \boxminus \mathbf{Z}_{ij}. \tag{3}$$

The minimization of cost function Eq. (1) is generally performed using ILS approaches - *e.g.* GN or Levenberg-Marquardt (LM). Since the problem is non-linear, those approaches solve a linear (or quadratic) approximation of the original problem around the current state configuration. The latter is obtained expanding the error Eq. (3) using the first-order Taylor expansion around $\breve{\mathbf{X}}$:

$$\begin{aligned}\mathbf{e}_{ij}(\breve{\mathbf{X}} \boxplus \mathbf{\Delta x}) &\approx \mathbf{e}_{ij}(\breve{\mathbf{X}}_i, \breve{\mathbf{X}}_j) + \left.\frac{\partial \mathbf{e}_{ij}(\breve{\mathbf{X}} \boxplus \mathbf{\Delta x})}{\partial \mathbf{\Delta x}}\right|_{\mathbf{\Delta x}=\mathbf{0}} \mathbf{\Delta x} \\ &= \mathbf{e}_{ij}(\breve{\mathbf{X}}_i, \breve{\mathbf{X}}_j) + \mathbf{J}_{ij}\mathbf{\Delta x}. \end{aligned} \tag{4}$$

Since the state space is $\mathrm{SE}(n)$ both for measurements and the states, we define an operator $\boxplus$ that applies an Euclidean perturbation $\mathbf{\Delta x}$ to a manifold object $\mathbf{X}$. Intuitively, if $\mathbf{\Delta x} = \mathbf{0}$ then $\mathbf{X} \boxplus \mathbf{\Delta x} = \mathbf{X}$. Introducing Eq. (4) each term of the cost function Eq. (1) represents a quadratic form in $\mathbf{\Delta x}$:

$$e_{ij}(\breve{\mathbf{X}} \boxplus \mathbf{\Delta x}) \approx \mathbf{\Delta x}^\top \mathbf{J}_{ij}^\top \mathbf{\Omega}_{ij} \mathbf{J}_{ij} \mathbf{\Delta x} + 2\mathbf{e}_{ij}^\top \mathbf{\Omega}_{ij} \mathbf{J}_{ij} \mathbf{\Delta x} + c.$$

Indicating with $\mathbf{H}_{ij} = \mathbf{J}_{ij}^\top \mathbf{\Omega}_{ij} \mathbf{J}_{ij}$ and with $\mathbf{b}^\top = \mathbf{e}_{ij}^\top \mathbf{\Omega}_{ij} \mathbf{J}_{ij}$ and extending the relation to all the graph's edges, we end in

the following:

$$\begin{aligned}\boldsymbol{\Delta x}^\star &= \underset{\boldsymbol{\Delta x}}{\operatorname{argmin}} \sum_{i,j} e_{ij}(\breve{\mathbf{X}} \boxplus \boldsymbol{\Delta x}) \\ &= \underset{\boldsymbol{\Delta x}}{\operatorname{argmin}} \boldsymbol{\Delta x}^\top \sum_{i,j} (\mathbf{H}_{ij}) \boldsymbol{\Delta x} + 2 \sum_{i,j} (\mathbf{b}_{ij}^\top) \boldsymbol{\Delta x} + c \\ &= \underset{\boldsymbol{\Delta x}}{\operatorname{argmin}} \boldsymbol{\Delta x}^\top \mathbf{H} \boldsymbol{\Delta x} + 2\mathbf{b}^\top \boldsymbol{\Delta x} + c \end{aligned} \tag{5}$$

In Eq. (5) we are computing the optimal state perturbation that will be applied to the current configuration through the previously defined operator $\bar{\mathbf{X}} = \breve{\mathbf{X}} \boxplus \boldsymbol{\Delta x}$. ILS algorithms repeatedly solve Eq. (5), until convergence is reached. We refer the reader to [3] for further details on Least-Squares based graph optimization. Summarizing, to perform graph optimization having both the nodes and edges lying on a smooth manifold we should define:

- proper parametrizations for the increment $\boldsymbol{\Delta x}$ and the error vector $\mathbf{e}_{ij}$
- the operator to apply an Euclidean perturbation to a manifold object - *i.e.* the $\boxplus$
- the operator to compute an Euclidean difference vector from two manifold objects - *i.e.* the $\boxminus$.

Once defined those entities, we can perform ILS optimization on arbitrary combinations of nodes and edges.

IV. Error Functions in SE(3)

In this scenario, the manifold representation for both nodes and measurements is a 3D isometry $\mathbf{T}$. A possible representation for the perturbation might be a 6D vector $\boldsymbol{\Delta x} = (\Delta x \ \Delta y \ \Delta z \ \Delta q_x \ \Delta q_y \ \Delta q_z)^\top$. Note that we indicate with $\mathbf{q} = (q_x \ q_y \ q_z)^\top$ the unit quaternion normalized by q_w, while $\mathbf{t} = (x \ y \ z)^\top$ describes the Cartesian coordinates in the space. Given this notation, the two parametrizations are:

$$\mathbf{X} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix} \quad \boldsymbol{\Delta x} = \begin{pmatrix} \boldsymbol{\Delta t}^\top & \boldsymbol{\Delta q}^\top \end{pmatrix}^\top. \tag{6}$$

In Eq. (6), $\mathbf{R} = \mathbf{R}(\mathbf{q})$ is the rotation matrix computed from the unit quaternion $\mathbf{q}$. Assuming that $\mathbf{e}_{ij} = (\mathbf{t}_e^\top \ \mathbf{q}_e^\top)^\top$ is represented in the same way of the perturbation, to perform the optimization we should define the operators $\boxplus$ and $\boxminus$ to apply the increment and compute the error vector. These are defined as follows:

$$\mathbf{X} \boxplus \boldsymbol{\Delta x} = \mathrm{v2t}(\boldsymbol{\Delta x})\mathbf{X} \qquad \mathbf{X}_a \boxminus \mathbf{X}_b = \mathrm{t2v}(\mathbf{X}_b^{-1}\mathbf{X}_a). \tag{7}$$

Given a perturbation vector $\boldsymbol{\Delta x}$ defined as in Eq. (6), v2t computes the relative isometry as follows:

$$\mathrm{v2t}(\boldsymbol{\Delta x}) = \begin{pmatrix} \mathbf{R} & \boldsymbol{\Delta t} \\ \mathbf{0}_{3\times 1} & 1 \end{pmatrix} \tag{8}$$

$$\boldsymbol{\Delta t} = \begin{bmatrix} \Delta x & \Delta y & \Delta z \end{bmatrix}^T \qquad \mathbf{R} = \mathbf{R}(\boldsymbol{\Delta q}) \tag{9}$$

On the contrary, the function t2v retrieves $\mathbf{t}_e$ and $\mathbf{q}_e$ from the isometry. Computing the translation from a SE(3) objected is a straightforward task, however, computing the minimal representation of the rotational component is not easy and involves several non-linear functions. A "smooth" function will have derivatives that change mildly and, thence, can be better approximated by its first-order Taylor expansion. On the contrary, less smooth functions might not be well approximated using only the first-order terms of the expansion, leading to a less stable optimization in general. In the remainder of this section we will propose the use of a different error function that is smoother and, thus, can be better described by its linear approximation.

A. Chordal-Distance Based Error Function

In this section we analyze a different error function for SE(3) objects based on the concept of *chordal distance* and we show how to embed it in the PGO context.

Given the manifold parametrization $\mathbf{X}$ in Eq. (6), we can define a simple function that given a 3D isometry $\mathbf{T}$, stacks its components in a 12D vector as follows:

$$\mathrm{flatten}(\mathbf{T}) = \begin{pmatrix} \mathbf{r}_1^\top & \mathbf{r}_2^\top & \mathbf{r}_3^\top & \mathbf{t}^\top \end{pmatrix}^\top \tag{10}$$

where $\mathbf{r}_k$ represents the k-th column of matrix $\mathbf{R}$. The inverse function is straightforwardly defined as:

$$\mathrm{unflatten}(\mathbf{v}) = \begin{bmatrix} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 \end{bmatrix} & \mathbf{t} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{11}$$

Hence, we redefine the $\boxminus$ as follows:

$$\begin{aligned}\mathbf{X}_a \boxminus_{ch} \mathbf{X}_b &= \mathrm{flatten}(\mathbf{X}_a) - \mathrm{flatten}(\mathbf{X}_b) \\ &= \mathrm{flatten}(\mathbf{X}_a - \mathbf{X}_b). \end{aligned} \tag{12}$$

In other words, in Eq. (12) we are computing the error vector through the element-wise difference between $\mathbf{X}_a$ and $\mathbf{X}_b$. As for the minimal parametrization for the increment, we use the Cartesian coordinates and Euler angles, resulting in $\boldsymbol{\Delta x} = [x \ y \ z \ \phi \ \theta \ \psi]^\top$. To pass from the increment parametrization to the extended one, we use the function $\mathrm{ev2t}(\boldsymbol{\Delta x})$. As a result the new operator $\boxplus_{euler}$ is defined as $\mathbf{X} \boxplus_{euler} \boldsymbol{\Delta x} = \mathrm{ev2t}(\boldsymbol{\Delta x})\mathbf{X}$. The complete mathematical derivation of the function ev2t is reported in the supplementary material. We can now introduce the new operators $\boxplus_{euler}$ and $\boxminus_{ch}$ in the standard optimization process presented in the previous sections. As a result, the error between predicted and actual measurement is computed as follows:

$$\mathbf{e}_{ij} = \hat{\mathbf{Z}}_{ij} \boxminus_{ch} \mathbf{Z}_{ij} = \mathrm{flatten}(\mathbf{X}_i^{-1}\mathbf{X}_j) - \mathrm{flatten}(\mathbf{Z}_{ij}). \tag{13}$$

The Jacobians $\mathbf{J}_i$ and $\mathbf{J}_j$ derived from Eq. (13) are much simpler to compute in closed form with respect to the standard error function. Moreover, the two Jacobians are linked by the relation $\mathbf{J}_i = -\mathbf{J}_j$. Thanks to this property, it is possible to speedup the composition of the linear system in the optimization step. We refer the reader to the supplementary material for further details on the Jacobians computation.

To keep the optimization consistent, the information matrix $\boldsymbol{\Omega}_{ij}$ relative to edge $\mathbf{Z}_{ij}$ should be projected onto the error space. This is generally done either through first-order error propagation or using the Unscented Transform. In our implementation, we preferred the latter, since it better handles the non-linearities. We report the complete derivation with the

Unscented Transform in the supplementary material. Note that our approach projects 6D points - *i.e.* the sigma points extracted from the measurement density - onto a 12D space. This means that the 12D reconstructed covariance $\bar{\mathbf{\Sigma}}_k^{-1}$ is rank deficient. Accordingly, we cannot exactly compute the information matrix $\bar{\mathbf{\Omega}}_k = \bar{\mathbf{\Sigma}}_k^{-1}$. We overcome this problem adding a value $\epsilon > 0$ to the null eigenvalues of $\bar{\mathbf{\Sigma}}_k$, before inverting it. We verified performing the inverse projection (from 12D to 6D) and checking that the result obtained is numerically close to the original. Nevertheless, we noticed that the right value of ϵ depends from the magnitude of the eigenvalues of $\mathbf{\Omega}_k$. One can also simply remove those null eigenvalues while computing $\bar{\mathbf{\Omega}}_k$, resulting, however, in a less stable optimization and in worse convergence properties in general. Further details about this parameter are reported in the next section.

V. Experimental Evaluation

In this section we propose a set of experiments to support our claim that the chordal error function in PGO is more robust to noise and has a larger convergence basin, compared to the standard error function that works in the unit-quaternion space. To this end, we performed convergence analysis of the optimization on several benchmark datasets for 3D PGO using both error functions. We varied both the noise figures applied to the measurements and the initial guess, to cover the majority of use cases. We compared the χ^2 evolution - *i.e.* the sum of the residual error $\mathbf{e}_{ij}$ computed on each measurement $\mathbf{Z}_{ij}$ - of the optimization in both cases, recomputing it at each iteration of the chordal optimization with the standard error function. In this way, we can obtain an unbiased evaluation of the convergence of the approaches. We will refer to the recomputed χ^2 as the *chordal reprojected* one. In the remaining of this section, all the plots of the chordal χ^2 actually refer to the *chordal reprojected* one. Furthermore, in the experiments we also report the evolution of the χ^2 feeding as input the ground truth. We do this to provide a baseline for χ^2 given the noise affecting the edges. Finally, in the supplementary material of this letter, we provide further experiments to validate our conjectures using alternative parametrizations both for the increments and the error vector.

A. Specifications of Datasets

We used standard 3D benchmark datasets for PGO which are publicly available. We generated also 3 more datasets, and those are released together with our g^2o plugin code. We applied different noise figures to the edges, resulting in multiple noise configurations. For each configuration, we sampled from $\mathcal{N}_t(\mathbf{0}, \mathbf{\Sigma}_t)$ and $\mathcal{N}_R(\mathbf{0}, \mathbf{\Sigma}_R)$ respectively for the translational and rotational components of the measurement. Experiments provided can be grouped in two, depending on the shape of the noise applied:

- Isotropic noise: in this group the noise covariances $\mathbf{\Sigma}_t$ is of composed as $\mathbf{\Sigma}_t = \mathrm{diag}(\eta\,\eta\,\eta)$; the same applies to $\mathbf{\Sigma}_R$ - with a different value on the diagonal.
- Anisotropic noise: we tried to simulate a planar navigation, a common task in generic Robotics application. As a result, the covariances relative to the translational and rotational components are respectively $\mathbf{\Sigma}_t = \mathrm{diag}(\eta\,\eta\,\zeta)$ with $\eta > \zeta$ and $\mathbf{\Sigma}_R = \mathrm{diag}(\xi\,\xi\,\kappa)$ with $\kappa > \xi$. Therefore, their shapes are highly non spherical, impacting on the convergence of the optimization process.

Further details on the noise distributions used in the experiments are reported in Tab. I, both for isotropic and anisotropic cases. We performed the tests on 10 random noise realizations for each distribution reported in Tab. I. Finally, we initialized the graphs using different techniques: odometry, the spanning-tree implementation of g^2o and the chordal initialization of Martinec *et al.* [32]. The remaining of this section is structured as follows: in Sec. V-B we will analyze the convergence of our approach in the case of isotropic noise, while in Sec. V-C we will consider anisotropic one; finally, we will investigate the effect of projecting the original 6D information matrix $\mathbf{\Omega}_{ij}$ onto a higher dimensional space. On the software web-page, we report the results for each noise scenario. However, due to space limitations, in this letter we show only the results obtained with severe noise figures, since the approaches are substantially equivalent in the case of small noise.

B. Isotropic Covariance

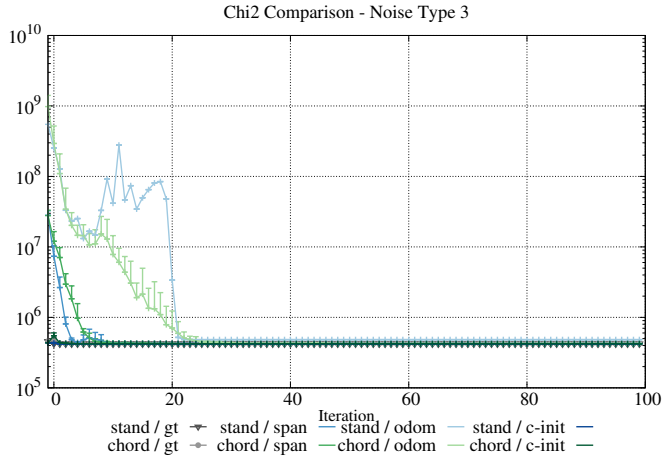
In this scenario, we projected edges information matrix $\mathbf{\Omega}_{ij}$ using $\epsilon = 0.1$. The optimization algorithm is the standard GN implemented in g^2o. Considering severe noise figures - *i.e.* `3A` - the increased smoothness of our error function leads to better results in general. In particular, both approaches fail in finding the optimal solution in many cases. Still, we noticed that the standard error function is less numerically stable, leading to fatal optimization failure in most of the trials when the graph is badly initialized. In Fig. 2a, we report the evolution of the χ^2 on the dataset `sim-manhattan`, where we consider only trials that do not incur in fatal errors. However, as reported in Fig. 2b, the standard error function leads to fatal errors in most of the trials when starting from bad initial guesses. Finally, Fig. 3 shows that the time to perform one iteration of GN - the standard g^2o implementation - remains similar to the standard approach, despite the fact that the matrices involved have bigger dimensions. The plot reports the mean time per-iteration together with its standard deviation, computed over 100 iterations.

C. Anisotropic Covariance

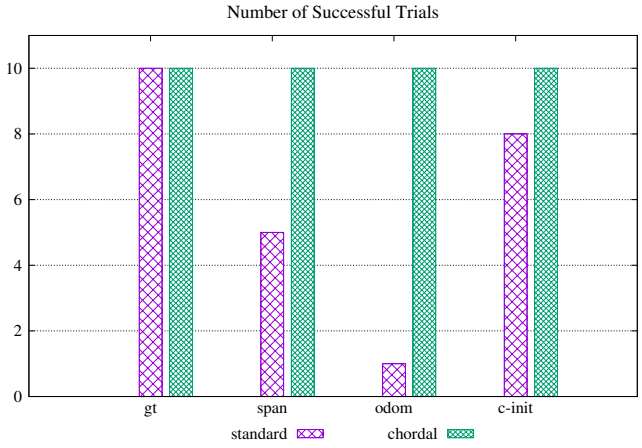
In these experiments, the uneven shape of the noise information matrix $\mathbf{\Omega}_{ij}$ requires a lower value for the conversion factor ϵ. The number of degenerate eigen values is higher and, therefore, using a value $\epsilon > 10^{-6}$ will lead to the solution of a different problem. We empirically found that $\epsilon = 10^{-6}$ is a good conversion value in such scenario. The optimization algorithm is again GN. Also in this case, when the noise is relatively bounded - *e.g.* type `1B` and `2B` - the two approaches are basically equivalent. They both reach the optimal configuration in few iterations, independently from the initial guess, in almost all the datasets. Nevertheless, increasing the

Isotropic			Anisotropic		
Config. Name	$\mathbf{\Sigma}_t$ $[m]$	$\mathbf{\Sigma}_R$ $[rad]$	**Config. Name**	$\mathbf{\Sigma}_t$ $[m]$	$\mathbf{\Sigma}_R$ $[rad]$
1A	$\text{diag}(0.01; 0.01; 0.01)$	$\text{diag}(0.001; 0.001; 0.001)$	1B	$\text{diag}(0.01; 0.01; 0.001)$	$\text{diag}(0.0001; 0.0001; 0.001)$
2A	$\text{diag}(0.1; 0.1; 0.1)$	$\text{diag}(0.01; 0.01; 0.01)$	2B	$\text{diag}(0.1; 0.1; 0.01)$	$\text{diag}(0.0001; 0.0001; 0.01)$
3A	$\text{diag}(0.5; 0.5; 0.5)$	$\text{diag}(0.1; 0.1; 0.1)$	3B	$\text{diag}(0.5; 0.5; 0.01)$	$\text{diag}(0.0001; 0.0001; 0.1)$

TABLE I: Covariances of the noise distribution used to perturb the measurements.



(a) Noise level 3A, dataset sim-manhattan.



(b) Number of successful optimization runs.

Fig. 2: Evolution of the χ^2 in the sim-manhattan dataset under noise of type 3A (left). We used GN as optimization algorithm, starting from multiple initial guesses - *i.e.* odometry, spanning tree and chordal initialization. As the baseline, we report the evolution of the χ^2 when the initial guess is the ground-truth. To fairly compare the two approaches, we only consider the trials that have not incurred in fatal errors. On the right: number of trials that succeeded in performing the optimization.

noise affecting the edges to level 3B produces different results. The extremely uneven shape of the information matrix $\mathbf{\Omega}_{ij}$ emphasizes the non-linearity of the standard error function, leading to sub-optimal solutions in several trials. Using the chordal error leads to better results in general, even from bad initial guesses. In Tab. II we report a comparison of the results obtained performing 100 iterations of GN with measurements affected by noise of type 3B. We used different initialization for the graph to analyze the convergence basin of the two error functions. We reported as baseline the χ^2 obtained after the optimization when the initial guess is the ground truth - referred as *Optimal* χ^2 in Tab. II. The analysis has been performed on 10 different noise realizations. From the latter we can also observe that the two minima are numerically close in all the cases. From these experiments the reader might notice how the standard error function is able to reach a better χ^2 mainly when the initial guess fed is not so far from the optimum - *i.e.* the chordal initialization. In badly initialized graph, the greater smoothness of the chordal error function helps the convergence, reaching better configurations - even if they are sub-optimal. In Tab. II, we report also the number of times that the optimization converged to a local or global optimum - starting from different initial guesses. Note that, there are some scenarios in which using the standard error function leads to failure in *all* the trials, while our approach leads to local minima not far from the optimum - *e.g.* dataset sim-manhattan using the odometry as guess. Times per iteration are almost identical to the ones reported in Sec. V-B and, thus, due to the space limitations of the manuscript, are not reported here.

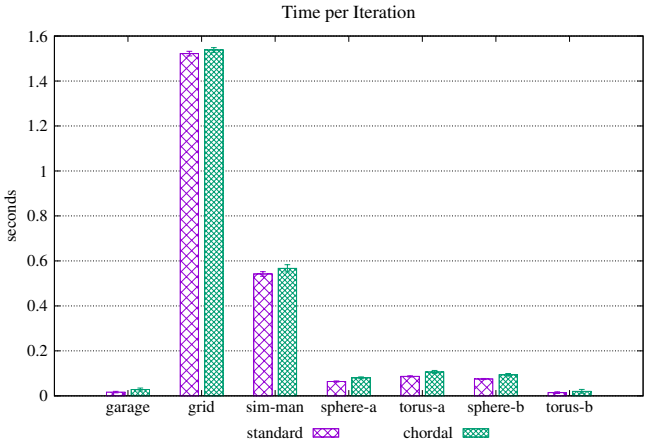


Fig. 3: Mean and standard deviation of the time per iteration using the vanilla g^2o implementation of GN. The initial guess is the ground-truth. Best in color.

D. Using Different Error and Perturbation Spaces

We conducted the previously presented experiments using the standard implementation of g^2o for the quaternion-based error function. In this case, the error vector is $\mathbf{e} = [e_x \;\; e_y \;\; e_z \;\; e_{qx} \;\; e_{qy} \;\; e_{qz}]^\top$, where $[e_{qx} \;\; e_{qy} \;\; e_{qz}]$ represents the rotation mismatch as unit quaternion. The same parametrization is also used for the increment vector $\mathbf{\Delta x} = [\Delta x_x \;\; \Delta x_y \;\; \Delta x_z \;\; \Delta x_{qx} \;\; \Delta x_{qy} \;\; \Delta x_{qz}]^\top$. Another commonly used minimal parametrization for $\text{SE}(3)$ objects is the one that uses Euler angles. Given that the parametrization adopted in the error vector might be different from the one of

	Guess	Initial χ^2	Final χ^2 [our]	Final χ^2 [stand.]	Optimum [our]	Optimum [stand.]
gar	odom	4.38E+10	7.75E+05 ± 198.7% [10]	- [0]	χ^2 = 4.58E+04	χ^2 = 3.43E+04
	span	1.68E+07	1.75E+06 ± 279.4% [10]	- [0]	ATE$_{pos}$ = 7.395 [m]	ATE$_{pos}$ = 7.679 [m]
	c-init	2.44E+06	8.66E+05 ± 81.1% [10]	8.05E+04 ± 1.7% [2]	ATE$_{rot}$ = 0.585 [rad]	ATE$_{rot}$ = 0.607 [rad]
grid	odom	4.64E+11	2.42E+09 ± 112.4% [10]	4.05E+09 ± 52.8% [10]	χ^2 = 1.01E+06	χ^2 = 9.66E+05
	span	2.67E+11	1.01E+06 ± 3.2% [10]	3.20E+10 ± 60.0% [10]	ATE$_{pos}$ = 1.666 [m]	ATE$_{pos}$ = 1.668 [m]
	c-init	4.06E+09	1.01E+06 ± 3.2% [10]	9.66E+05 ± 3.0% [10]	ATE$_{rot}$ = 0.032 [rad]	ATE$_{rot}$ = 0.025 [rad]
s-man	odom	2.35E+10	7.38E+05 ± 32.0% [10]	- [0]	χ^2 = 4.31E+05	χ^2 = 3.85E+05
	span	5.77E+07	4.31E+05 ± 0.5% [10]	3.86E+05 ± 1.5% [6]	ATE$_{pos}$ = 0.960 [m]	ATE$_{pos}$ = 0.816 [m]
	c-init	1.10E+06	4.31E+05 ± 0.5% [10]	3.85E+05 ± 0.3% [10]	ATE$_{rot}$ = 0.227 [rad]	ATE$_{rot}$ = 0.171 [rad]
sph-b	odom	3.82E+11	1.47E+05 ± 1.4% [10]	5.98E+09 ± 150.7% [10]	χ^2 = 1.47E+05	χ^2 = 1.23E+05
	span	4.60E+11	1.47E+05 ± 1.4% [10]	5.00E+09 ± 53.8% [10]	ATE$_{pos}$ = 1.208 [m]	ATE$_{pos}$ = 1.223 [m]
	c-init	1.20E+09	1.06E+08 ± 14.7% [10]	7.22E+09 ± 245.2% [10]	ATE$_{rot}$ = 0.071 [rad]	ATE$_{rot}$ = 0.048 [rad]
tor-b	odom	2.01E+09	1.02E+08 ± 187.2% [10]	4.95E+11 ± 137.8% [3]	χ^2 = 1.19E+04	χ^2 = 8.75E+03
	span	1.54E+10	1.57E+06 ± 298.7% [10]	2.52E+12 ± 173.0% [8]	ATE$_{pos}$ = 1.116 [m]	ATE$_{pos}$ = 0.967 [m]
	c-init	6.48E+07	1.19E+04 ± 8.6% [10]	8.75E+03 ± 2.9% [10]	ATE$_{rot}$ = 0.229 [rad]	ATE$_{rot}$ = 0.166 [rad]
sph-a	odom	3.79E+10	5.07E+06 ± 163.5% [10]	1.67E+09 ± 79.0% [10]	χ^2 = 2.37E+05	χ^2 = 1.58E+05
	span	1.43E+10	2.18E+06 ± 267.0% [10]	3.35E+06 ± 186.0% [10]	ATE$_{pos}$ = 1.380 [m]	ATE$_{pos}$ = 1.357 [m]
	c-init	4.64E+07	2.37E+05 ± 2.3% [10]	1.58E+05 ± 2.1% [10]	ATE$_{rot}$ = 0.109 [rad]	ATE$_{rot}$ = 0.062 [rad]
tor-a	odom	1.74E+11	1.07E+09 ± 20.0% [10]	- [0]	χ^2 = 3.99E+04	χ^2 = 3.96E+04
	span	5.20E+10	3.96E+07 ± 61.0% [10]	- [0]	ATE$_{pos}$ = 1.143 [m]	ATE$_{pos}$ = 1.143 [m]
	c-init	1.95E+09	3.99E+04 ± 1.5% [10]	3.96E+04 ± 1.6% [9]	ATE$_{rot}$ = 0.037 [rad]	ATE$_{rot}$ = 0.034 [rad]

TABLE II: Analysis of the χ^2 on multiple datasets. The noise added is of type 3B. Experiments are performed over a population of 10 noise trials. Column 3 reports the mean χ^2 of the initial guess. Column 4 and 5 contain mean and standard deviation of the final χ^2 obtained after 100 iterations of GN optimization. Between squared brackets we indicate the number of trials that converged to a local or global optimum. Finally, the last 2 columns contain mean values of i) the χ^2 , ii) the translational and iii) rotational Absolute Tracjectory RMSE computed after 100 GN iterations starting from the ground-truth.

the increments, one can mix quaternion-based and Euler-based parametrization to describe error and increments. In the supplementary material, we report further analysis on the χ^2 aimed to analyze how changing from one parametrization to the other affects the convergence. In particular, we used our error function in conjunction with quaternion *and* Euler based increments; as for the standard error function, we used both the quaternion *and* the Euler based representation for the error vector, combined with quaternion *and* Euler based increments. These tests confirm the trend highlighted in Sec. V-B and Sec. V-C.

E. Impact of Information Matrix Projection

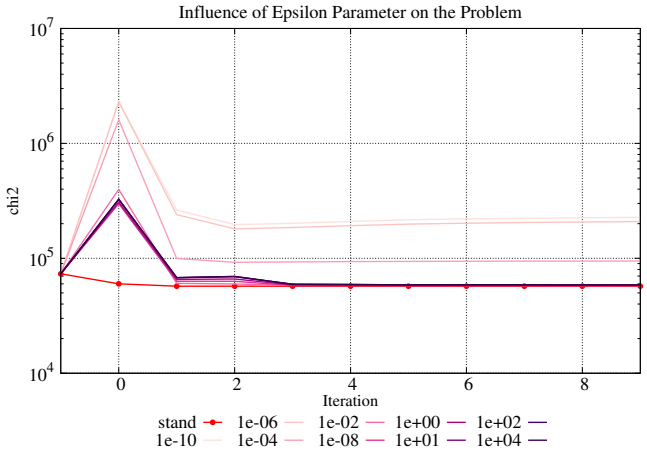
In Sec. IV-A we explained that the error space using the chordal error function has dimension 12 and, thus, we have to project onto it the original information matrix $\mathbf{\Omega}_{ij}$. A possible way to achieve this goal in relatively easy way is using the Unscented Transform. We sample sigma points from the measurement distribution $\mathcal{N}(\mathbf{z}_{ij}, \mathbf{\Sigma}_{ij})$ - where $\mathbf{z}_{ij} = \mathrm{t2v}(\mathbf{Z}_{ij})$ and $\mathbf{\Omega}_{ij} = \mathbf{\Sigma}_{ij}^{-1}$ - and project them onto the new higher dimensional space. In this way, we can compute the mean and covariance of the new distribution $\mathcal{N}(\bar{\mathbf{z}}_{ij}, \bar{\mathbf{\Sigma}}_{ij})$. The converted information matrix is simply the inverse of $\bar{\mathbf{\Sigma}}_{ij}$.

Nonetheless, since we are projecting 6D sigma points onto a 12D space, the resulting covariance may lose rank multiple times. As a result the hyper-ellipsoid described by $\bar{\mathbf{\Sigma}}_{ij}$ may have many degenerated eigenvalues - as many as the difference between the space dimension and the current rank of $\bar{\mathbf{\Sigma}}_{ij}$. Intuitively, if $\bar{\mathbf{\Sigma}}_{ij}$ is already oddly shaped - as in our case - the number of degenerated eigenvalues of the projection rises.

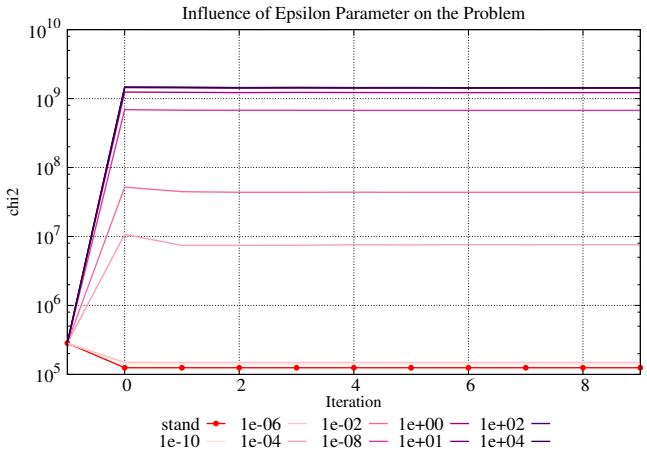
To properly invert $\bar{\mathbf{\Sigma}}_{ij}$ and perform the optimization process correctly, we sum a value $\epsilon > 0$ to the eigenvalues of $\bar{\mathbf{\Sigma}}_{ij}$ that are below a threshold, retrieving the information matrix as $\bar{\mathbf{\Omega}}_{ij} = (\mathbf{U}\hat{\mathbf{S}}\mathbf{V}^\top)^{-1}$, where $\hat{\mathbf{S}}$ is the matrix containing the conditioned eigenvalues. When $\bar{\mathbf{\Sigma}}_{ij}$ is subjected to multiple rank losses, conditioning it with $\epsilon > 10^{-6}$ conducts to a different problem formulation, more relaxed. Therefore the optimal solution - as the final residual error - is different from the original problem. In Fig. 4 we reported the residual error of the optimal configuration varying the value of ϵ. We used the sphere-b dataset, with noise figures 3A and 3B, representing respectively isotropic and anisotropic information matrix cases. In this experiment, the initial guess is the ground truth. As a result, uneven shapes of $\mathbf{\Omega}_{ij}$ require smaller values of ϵ to achieve the equivalence of the problem. In cases where $\mathbf{\Omega}_{ij}$ has a more spherical appearance, larger value of ϵ are preferred to ensure the convergence to the global minimum and to guarantee that the optimal solution is equivalent to the original. Still, visual inspection of solutions that report a different χ^2 compared to the original problem, does not reveal any notable difference.

VI. Conclusion

In this work, we analyzed the effects of a different error function to be used in 3D PGO. The proposed function is based on the concept matrix difference between isometries and compared to the gold standard approach for PGO - *i.e.* that computes the error in the unit-quaternion space - reports several benefits. The greater smoothness of the function allows being better approximated through its first-order Taylor expansion, leading to an enlarged convergence basin. Furthermore, its derivatives are rather simple to compute in close form and have good numerical properties. Theoretically, this also allows reducing the number of FLOPS required to build the linear system. We performed several experiments to support our claims, varying both the noise figure super-imposed on the edges and the initialization technique. We inspected also the effects of anisotropic measurements covariance matrices on the optimization process, reporting a better trend when using our approach. Finally, we analyzed whether the converted chordal



(a) Nearly-spherical $\mathbf{\Omega}_{ij}$.



(b) Non-spherical $\mathbf{\Omega}_{ij}$.

Fig. 4: Comparison of the chordal reprojected χ^2 for different values of ϵ - best in color. The initial guess is the ground truth, while the dataset is `sphere-b` in all runs. This experiment considers only on 1 noise sample.

problem is equivalent to the original one. As a result, the two approaches reach basically the same global minimum under realistic noise figures.

REFERENCES

[1] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

[2] T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (slam): Part ii. *IEEE Robotics & Automation Magazine*, 13(3):108–117, 2006.

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

[4] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and Wolfram Burgard. g 2 o: A general framework for graph optimization. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 3607–3613. IEEE, 2011.

[5] F. Dellaert. Factor graphs and gtsam: A hands-on introduction. Technical report, Georgia Institute of Technology, 2012.

[6] Agarwal S., Mierle K., and Others. Ceres solver. http://ceres-solver.org.

[7] L. Carlone, R. Tron, K. Daniilidis, and F. Dellaert. Initialization techniques for 3d slam: a survey on rotation estimation and its use in pose graph optimization. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 4597–4604. IEEE, 2015.

[8] MWM G. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (slam) problem. *IEEE Trans. on Robotics and Automation*, 17(3):229–241, 2001.

[9] A. J. Davison and D. W. Murray. Simultaneous localization and map-building using active vision. *IEEE transactions on pattern analysis and machine intelligence*, 24(7):865–880, 2002.

[10] J. Leonard and P. Newman. Consistent, convergent, and constant-time slam. In *Proc. of the Intl. Conf. on Artificial Intelligence (IJCAI)*, pages 1143–1150, 2003.

[11] S. Se, D. Lowe, and J. Little. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *Intl. Journal of Robotics Research (IJRR)*, 21(8):735–758, 2002.

[12] J. A Castellanos, J. Neira, and J. D. Tardós. Limits to the consistency of ekf-based slam. *IFAC Proceedings Volumes*, 37(8):716–721, 2004.

[13] L. A. Clemente, A. J. Davison, I. D. Reid, J. Neira, and J. D. Tardós. Mapping large loops with a single hand-held camera. In *Proc. of Robotics: Science and Systems (RSS)*, volume 2, 2007.

[14] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, et al. Fastslam: A factored solution to the simultaneous localization and mapping problem. *Proc. of the Conference on Advancements of Artificial Intelligence (AAAI)*, 593598, 2002.

[15] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, et al. Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proc. of the Intl. Conf. on Artificial Intelligence (IJCAI)*, pages 1151–1156, 2003.

[16] G. Grisetti, C. Stachniss, W. Burgard, et al. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Trans. on Robotics (TRO)*, 23(1):34, 2007.

[17] S. S. Haykin and S. S Haykin. *Kalman filtering and neural networks*. Wiley Online Library, 2001.

[18] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous robots*, 4(4):333–349, 1997.

[19] J.S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *Computational Intelligence in Robotics and Automation, 1999. CIRA'99. Proceedings. 1999 IEEE International Symposium on*, pages 318–325. IEEE, 1999.

[20] T. Duckett, S. Marsland, and J. Shapiro. Learning globally consistent maps by relaxation. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, volume 4, pages 3841–3846. IEEE, 2000.

[21] U. Frese, P. Larsson, and T. Duckett. A multilevel relaxation algorithm for simultaneous localization and mapping. *IEEE Trans. on Robotics (TRO)*, 21(2):196–207, 2005.

[22] A. Howard, M.J. Mataric, and G. Sukhatme. Relaxation on a mesh: a formalism for generalized localization. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, volume 2, pages 1055–1060. IEEE, 2001.

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

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

[25] M. T. Lázaro, R. Capobianco, and G. Grisetti. Efficient long-term mapping in dynamic environments. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 153–160. IEEE, 2018.

[26] F. Dellaert and M. Kaess. Square root sam: Simultaneous localization and mapping via square root information smoothing. *Intl. Journal of Robotics Research (IJRR)*, 25(12):1181–1203, 2006.

[27] M. Kaess, A. Ranganathan, and F. Dellaert. isam: Fast incremental smoothing and mapping with efficient data association. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 1670–1677. IEEE, 2007.

[28] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert. isam2: Incremental smoothing and mapping using the bayes tree. *Intl. Journal of Robotics Research (IJRR)*, 31(2):216–235, 2012.

[29] K. Ni and F. Dellaert. Multi-level submap based slam using nested dissection. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 2558–2565. IEEE, 2010.

[30] G. Grisetti, R. Kümmerle, and K. Ni. Robust optimization of factor graphs by using condensed measurements. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 581–588. IEEE, 2012.

[31] V. Ila, L. Polok, M. Solony, and P. Svoboda. Slam++-a highly efficient and temporally scalable incremental slam framework. *Intl. Journal of Robotics Research (IJRR)*, 36(2):210–230, 2017.

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

[33] M. Cucuringu, Y. Lipman, and A. Singer. Sensor network localization by eigenvector synchronization over the euclidean group. *ACM Transactions on Sensor Networks (TOSN)*, 8(3):19, 2012.

[34] F. Arrigoni, B. Rossi, and A. Fusiello. Spectral synchronization of multiple views in se (3). *SIAM Journal on Imaging Sciences*, 9(4):1963–1990, 2016.

[35] 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. *Intl. Journal of Robotics Research (IJRR)*, page 0278364918784361, 2016.

[36] J. Briales and J. Gonzalez-Jimenez. Cartan-sync: Fast and global se (d)-synchronization. *IEEE Robotics and Automation Letters (RA-L)*, 2(4):2127–2134, 2017.

[37] H. Wang, S. Huang, G. Yang, and G. Dissanayake. Comparison of two different objective functions in 2d point feature slam. *Automatica*, 97:172–181, 2018.

[38] L. Carlone, D. M. Rosen, G. Calafiore, J. J. Leonard, and F. Dellaert. Lagrangian duality in 3d slam: Verification techniques and optimal solutions. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 125–132. IEEE, 2015.

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