COP-SLAM: Closed-Form Online Pose-Chain Optimization for Visual SLAM

Gijs Dubbelman, *Member, IEEE*, and Brett Browning, *Member, IEEE*

***Abstract*—In this paper, we analyze and extend the recently proposed closed-form online pose-chain simultaneous localization and mapping (SLAM) algorithm. Pose-chains are a specific type of extremely sparse pose-graphs and a product of contemporary SLAM front-ends, which perform accurate visual odometry and reliable appearance-based loop detection. They are relevant for challenging robotic applications in large-scale 3-D environments for which frequent loop detection is not desired or not possible. Closed-form online pose-chain SLAM efficiently and accurately optimizes pose-chains by exploiting their Lie group structure. The convergence and optimality properties of this solution are discussed in detail and are compared against state-of-the-art iterative methods. We also provide a novel solution space, that of similarity transforms, which has not been considered earlier for the proposed algorithm. This allows for closed-form optimization of pose-chains that exhibit scale drift, which is important to monocular SLAM systems. On the basis of extensive experiments, specifically targeting 3-D pose-chains and using a total of 60 km of challenging binocular and monocular data, it is shown that the accuracy obtained by closed-form online pose-chain SLAM is comparable with that of state-of-the-art iterative methods, while the time it needs to compute its solution is orders of magnitudes lower. This novel SLAM technique thereby is relevant to a broad range of robotic applications and computational platforms.**

***Index Terms*—Computer vision, pose-graph optimization, simultaneous localization and mapping (SLAM).**

I. Introduction

We focus on the challenge of simultaneous localization and mapping (SLAM) [1] solely on the basis of visual sensors and both analyze and extend a recently proposed solution called *closed-form online pose-chain SLAM* (COP-SLAM) [2]. Improving on current SLAM solutions is relevant to many contemporary and future robotic applications such as intelligent vehicles, autonomous inspection and surveying platforms, as well as supportive healthcare and domestic robots. The current consensus [3] is to use filter-based solutions, e.g., based on EKF-SLAM, UKF-SLAM, IKF-SLAM [4], [5] or particle filter SLAM [6]–[8], when computational resources are limited, and to use more accurate graph-based approaches [9]–[16], bundle adjustment-based approaches [17]–[19], or a combination [20] when sufficient computational resources are available. COP-SLAM optimizes a specific kind of extremely sparse graphs, called pose-chains, and its contribution is that it computes a nonlinear solution in closed form for large-scale 3-D SLAM problems. Although COP-SLAM can also be used to optimize (or initialize) dense pose-graphs, its advantages are most relevant when it is applied to pose-chains. The conceptual differences between general graph-based SLAM, pose-graph SLAM, and pose-chain SLAM is discussed in Section I-A.

SLAM systems using graph-based representations typically consist of front-end and back-end subsystems. The responsibility of the front-end is to provide an initial solution, e.g., by using odometry, visual odometry [21]–[23], or structure from motion [24], [25] methods, as well as to detect loops in the robot's trajectory, e.g., by using appearance-based loop detection [26], [27] or local map matching [28]. The responsibility of the back-end is to turn the initial solution into a maximum likelihood solution considering all available data. Such approaches are typically more accurate than filter-based approaches, as they are better able to capture the nonlinearities that are inherent to SLAM. At their core, graph-based approaches, of which bundle adjustment is a specific example, rely on computationally demanding nonlinear iterative optimization techniques with their usual sensitivity to improper initialization.

COP-SLAM is specifically designed to be a lightweight SLAM back-end that is to be used with strong vision-based SLAM front-ends, which perform accurate visual odometry and reliable appearance-based loop detection. The output of such contemporary SLAM front-ends in large-scale environments are chains of many relative poses with relatively few loop-closure poses. These pose-chains are optimized by COP-SLAM in closed form while respecting most of their nonlinearities, by employing Lie group methods. COP-SLAM can be used with any Lie group for which its logarithmic map and exponential map can be computed; it can, therefore, be applied to 2-D and to 3-D SLAM problems. The ability to use closed-form nonlinear optimization for 3-D pose-chains is currently a unique and novel theoretical property offered by COP-SLAM. In Section II, the inner workings of COP-SLAM are described in detail, and a theoretical analysis of its convergence and optimality properties is provided. This analysis is one of the main contributions of this study. The relation to other efficient graph optimizers is described in Section I-B.

COP-SLAM was first presented in [2] and is based on earlier work on trajectory bending [29], [30], which in turn was inspired by Newman [31]. Until now, it has only been applied to

Manuscript received September 29, 2014; revised April 7, 2015; accepted August 23, 2015. Date of publication September 18, 2015; date of current version September 30, 2015. This paper was recommended for publication by Associate Editor D. Scaramuzza and Editor D. Fox upon evaluation of the reviewers' comments.

G. Dubbelman is with the Department of Electrical Engineering, Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands (e-mail: g.dubbelman@tue.nl).

B. Browning is with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: brettb@cs.cmu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2015.2473455

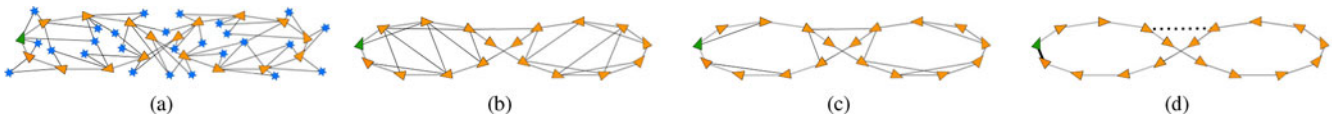

Fig. 1. Different types of graph-based SLAM. (a) General graph SLAM, (b) pose-graph SLAM, (c) sparse pose-graph SLAM, and (d) pose-chain SLAM. The initial absolute pose is depicted by the green triangle and all successive absolute poses by orange triangles. Landmarks are shown by blue stars. In (a), the nodes in the graph model absolute poses and landmarks. The edges, depicted by the black lines, between the nodes in (a) model landmark observations. In (b) and (c), the nodes in the pose-graph only model absolute poses and edges model the relative poses between them. This is the same for the pose-chain depicted in (d). In pose-chain SLAM, we, however, only model edges between successive nodes and (loop-closing) edges that contain sufficient information on the globally accumulated error of the nodes. In COP-SLAM, only the most recent loop closure is considered as an active edge; it is depicted by the heavy black line in (d). Previous loop closures, e.g., the one depicted by the dashed heavy black line in (d), that have been processed earlier, are no longer actively considered, but are accounted for by a filter-like mechanism inside COP-SLAM. Iterative optimizers consider all edges of current and past loop closures as active edges.

pose-chains in which each pose is represented by a six-dimension-of-freedom (dof) Euclidean motion, i.e., an element of the Lie group SE(3). A second important contribution of this study is that we extend COP-SLAM with a novel solution space, i.e., the space of seven-dof similarity transforms SIM(3). This has great practical value when applying COP-SLAM to pose-chains that are obtained from monocular odometry where no constraints can be imposed to prevent scale drift. From the theoretical analysis in Section II, it will become clear that the optimality properties of COP-SLAM partially depend on the Lie group properties of the solution space to which it is applied. Both solution spaces SE(3) and SIM(3) are, therefore, discussed in Section III.

In Section IV, we investigate to which extent the theoretical differences between COP-SLAM and its iterative counterparts influence their performance on challenging large-scale 3-D pose-chain datasets. We use simulated data for detailed analysis as well as extensive real-world binocular and monocular datasets to highlight the applied benefits of COP-SLAM. The pose-chains used for our experiments as well as our COP-SLAM implementation are made publicly available in [32]. In Section IV, we also provide our third contribution, by incorporating landmark position updating into COP-SLAM, and show that this provides an excellent closed-form initialization strategy to sequential bundle adjustment. Finally, our conclusions are put forward in Section V.

A. *Pose-Graph Simultaneous Localization and Mapping Versus Pose-Chain Simultaneous Localization and Mapping*

The conceptual differences between general graph-based SLAM, pose-graph SLAM, and pose-chain SLAM are depicted in Fig. 1 and described further in this section.

About a decade ago, the SLAM front-end typically consisted of regular odometry (wheel rotation and steering angle) with, for example, a single laser range scanner. Such front-ends produced relatively erroneous initial estimates, creating the need to optimize with respect to poses and landmarks within the SLAM back-end [see Fig. 1(a)]. As sensor technology and processing power advanced over the years, the methods used inside front-ends started to produce more accurate initial estimates. This lowered the requirements given to the back-end, which made pose-graph SLAM a popular technique [see Fig. 1(b)]. With the increase of the accuracy of front-ends, the need to keep track of all edges between nodes diminished, allowing for sparse pose-graph approaches [see Fig. 1(c)]. It is important to consider that the reduction of the number of edges in sparse pose-graphs is not necessarily the result of an explicit marginalization or sparsification process acting on a dense graph. It can also be a direct consequence of the type of SLAM front-end that is being used, the robot's exploration strategy, or challenges posed by the robot's environment. This is illustrated in Fig. 2.

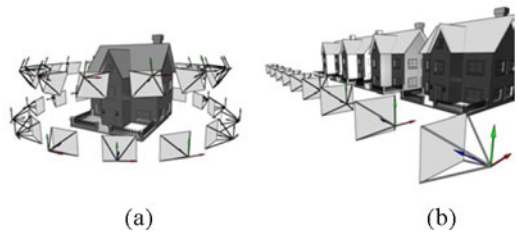

Fig. 2. Connectivity of a graph in graph-based SLAM is partially determined by the observation processing used in the front-end and partially determined by the robot's environment and the robot's movement. In (a), the robot is moving around a single object and always viewing this object. Such a setting results in dense graphs, as landmarks can be tracked over many views. In (b), a more realistic robot movement is depicted, which is typical for a robot moving through a street or through a corridor. Such a setting results in relatively sparser graphs, as most landmarks can only be tracked over relatively fewer frames.

Visual odometry and visual loop detection methods have significantly gained in accuracy and robustness in the past decade. Contemporary methods based on RANSAC [33], [34] and sliding-window bundle adjustment (possibly aided by inertial measurement units) are able to robustly estimate the robot's pose with high accuracy over significant distances [35]. As these vision-based front-ends can themselves optimize over multiple frames and only accumulate error (drift) slowly, there is no direct necessity to model dense dependences between poses; moreover, loop closures need to be performed significantly less times than when using a regular odometer. The ability to stay accurately localized while requiring less loop closures allows robots to perform more efficient and effective exploration strategies in more diverse and more large-scale environments. These developments allow for a shift of using a relatively weak regular odometry front-end, which has to be compensated for by using a strong back-end, to systems that use a strong vision-based front-end together with a lightweight back-end.

COP-SLAM is such a lightweight back-end and is specifically designed to be used with strong vision-based front-ends. It models the robot's trajectory with extremely sparse pose-graphs, called pose-chains. In pose-chains, *successive edges*, i.e., relative poses from time t to $t+1$, are added to the chain and directly provide adequate initial estimates for the nodes, i.e., the absolute poses. In the context of this study, we specifically define pose-chains to have high *local accuracy*, but their *global*

accuracy can be arbitrarily low. While this is a valid property for slowly drifting contemporary vision-based front-ends, there are clearly no such restrictions for pose-graphs in general.

The process of adding successive edges to the pose-chain continues until a loop is detected by the SLAM front-end. Loop detection (and verification) adds a *loop-closing edge* to the chain, which links the final node of time t back to a node with time $t - l$, e.g., for a relatively large $l > 1000$. In COP-SLAM, the loop-closing edge is in some way special, as it contains vital information on the globally accumulated error of the nodes. In its purest form, a pose-chain can thus be seen as the sparsest possible pose-graph, which has many successive edges and only one loop-closing edge. As more successive edges and more loop-closing edges are added to the pose-chain over time, it turns into an increasingly denser but still an extremely sparse timely ordered pose-graph. It is important to consider that COP-SLAM does not discard or ignore loops that are detected by the front-end. It uses all detected loops, but those loops for which l is relatively large are most important to COP-SLAM.

As pose-chains are a type of pose-graphs, they can be optimized with general nonlinear iterative graph optimizers like [9]. Their extremely sparse structure, their high local accuracy, and the fact that they can be modeled as chains of Lie group transformations also allow using the specialized optimization methods of COP-SLAM. These are highly efficient and accurate, but it will also become clear that COP-SLAM does not share all advantages of general graph optimizers. COP-SLAM must, therefore, be seen as a highly relevant, but also a very specific, and specialized SLAM approach.

B. Related Work

A significant body of SLAM research is available on efficient graph optimization. Here, we only discuss recent work that is most relevant to the optimization approach used in COP-SLAM.

Carlone *et al.* [36] presented a closed-form optimization technique, called linear approximation for pose graph optimization (LAGO). It exploits a specific relative orientation representation, allowing the pose-graph problem to be formulated as a quadratic optimization task. It is then solved in three steps, involving only closed-form linear methods. Similar to COP-SLAM, optimizing for rotation and translation is performed separately. This is shown to be very efficient, even for extremely large pose-graphs, while the impact on accuracy with respect to iterative solvers is marginal. The critical difference between LAGO and COP-SLAM is that the applicability of the optimization strategy of LAGO is restricted to planar 2-D SLAM problems, whereas COP-SLAM can be applied to 2-D and 3-D SLAM problems. In fact, COP-SLAM can be applied to any (SLAM) problem in which the graph's nodes and edges can be modeled as Lie group transformations, for which Exponential and Logarithmic maps can be computed.

Using a carefully chosen (manifold-based) relative parameterization, as done by LAGO and by COP-SLAM, is a frequently used technique to make the SLAM problem less nonlinear. This can greatly reduce the number of iterations required by iterative optimizers and also improve their stability. For example, Olson *et al.* [14] presented an efficient technique for 2-D pose-graph SLAM based on a relative parameterization. Grisetti *et al.* [10] extended this approach to 3-D pose-graph SLAM. A similar concept is also used for bundle adjustment by Mei *et al.* [17]. All these methods are, however, iterative nonlinear methods and are not as efficient as closed-form methods like LAGO and COP-SLAM. Recently, Zhao *et al.* [37] proposed a submap relative parameterization in which submaps can be merged with linear optimization. Similar to COP-SLAM, the benefit of this approach is its stability and robustness, but in contrast with COP-SLAM, its optimized implementation is not more efficient than contemporary nonlinear iterative graph optimizers, like that of Grisetti *et al.* [9], for large-scale 3-D pose-chains.

Much research has focused on mechanisms to distinguish informative edges from less informative edges in the graph. Computation is then only spent on informative edges, while ignoring or removing less informative edges. A recent approach is the double-window-based optimization technique of Strasdat *et al.* [20] that, similar to COP-SLAM, is aimed at large-scale 3-D visual-SLAM tasks. This approach is based on the observation that keeping track of all edges between poses and landmarks is only really advantageous when the same landmarks are observed from many different poses. An example situation in which this happens is shown in Fig. 2(a). In many realistic scenarios, this does not always occur, as is shown in Fig. 2(b). Therefore, Strasdat *et al.* [20] use an approach that applies bundle-adjustment to densely connected parts of the graph and pose-graph SLAM to weakly connected parts all in one monolithic iterative nonlinear optimizer. The decision on what is densely connected and what is weakly connected is performed by a dynamic and fully automated process. The reasoning behind this is similar to that of COP-SLAM, but we *a priori* model everything as an extremely sparse pose-graph, i.e., a pose-chain. In theory, COP-SLAM can, therefore, never obtain the same levels of accuracy as [20]. However, in Section IV, we show that COP-SLAM does obtain satisfactory accuracy on realistic and challenging large-scale 3-D visual-SLAM datasets at a fraction of the computational footprint of iterative nonlinear methods.

In general, SLAM literature offers three established conceptual mechanisms that can be used to make pose-graph SLAM more stable and more efficient: 1) using a carefully chosen (manifold-based) relative parameterization; 2) using this parameterization to separately optimize for rotations and for translations; and 3) only optimize with respect to the most informative edges in the graph. COP-SLAM applies all these three conceptual mechanisms in novel ways, making it a unique SLAM method from a theoretical perspective. From an applied perspective, the benefits of COP-SLAM as a back-end optimizer cannot be seen separately from the use of slowly drifting vision-based front-ends. Such front-ends are currently very popular in visual-SLAM research, and their components are also publicly available [27], [38].

II. Closed-Form Online Pose-Chain Simultaneous Localization and Mapping

In this section, we provide a detailed theoretical description of COP-SLAM, which optimizes pose-chains online and in closed form. The ability to do this comes at the cost of not being able to

provide maximum likelihood solutions under the same general conditions as iterative optimizers. The underlying fundamental properties are derived here, and their applied effects are later on investigated with experiments in Section IV.

The conceptual working of COP-SLAM's closed-form solution is best introduced by considering how a single-loop pose-chain is optimized by an iterative nonlinear graph optimizer like [9]. A typical (nonlinear summed squared error) objective function for general pose-graph SLAM is

$$f(\mathrm{A}_1,\ldots,\mathrm{A}_n) = \sum_{i=2}^{n} \varepsilon(\mathrm{A}_{i-1},\mathrm{A}_i,\mathrm{M}_{i-1,i},\Sigma_{i-1,i})^2 + \sum_{j=1}^{l} \varepsilon(\mathrm{A}_k,\mathrm{A}_m,{}^j\mathrm{L}_{k,m},{}^j\Sigma_{k,m})^2. \quad (1)$$

Here, A_i is a node that models the absolute pose at time i, and $\mathrm{M}_{i-1,i}$ is an edge that models the relative pose that links the nodes of times $i-1$ and i. The jth (loop-closing) edge that links nodes of times steps k and m with $m-k>1$ is denoted with ${}^j\mathrm{L}_{k,m}$. The (nonlinear) function $\varepsilon(\mathrm{A}_i,\mathrm{A}_j,\mathrm{M}_{i,j},\Sigma_{i,j})$ puts an error metric on the difference between the relative pose of an edge $\mathrm{M}_{i,j}$ and the absolute poses of its two nodes A_i and A_j, taking into consideration the uncertainty in the edge, which is denoted with $\Sigma_{i,j}$. For our purposes, we have split the objective function in terms related to edges for which their time difference is unity (i.e., successive edges) and edges for which their time difference is larger than unity (i.e., loop-closing edges), but there is no fundamental reason to do so.

The task of the SLAM back-end is to minimize f with respect the absolute poses $\mathrm{A}_1,\ldots,\mathrm{A}_n$, which generally requires nonlinear iterative optimization. In case of a pure pose-chain, there is only one edge whose time difference is larger than unity. This is the loop-closing edge, and all other edges are successive edges. The interesting observations now is that after initialization (concatenating relative poses to form absolute poses), but before optimization, all the contributions of all error terms related to successive edges are 0 (because $\mathrm{A}_{i-1}^{-1}\mathrm{A}_i=\mathrm{M}_{i-1,i}$), and the total error is completely determined by the contribution of the single loop-closing edge ${}^1\mathrm{L}_{m,n}$. After nonlinear iterative optimization, the error of the loop-closing edge ${}^1\mathrm{L}_{m,n}$ is distributed optimally over all successive edges.

This concept, of distributing the error of a single loop-closing edges over successive edges, forms the basis for the closed-form solution of COP-SLAM, whose conceptual working is provided in Algorithm 1. At its core, COP-SLAM performs trajectory bending every time a loop is detected by the SLAM front-end. Trajectory bending acts on a single loop and ensures that this loop is closed. It only considers the current loop-closing edge and not any previous loop-closing edges. Hence, once a loop-closing edge is processed, it is no longer considered as an (active) edge in the pose-chain (but it can always be reactivated for further iterative optimization). COP-SLAM does, however, uses a filter-like mechanism, which allows it to account for the statistical information of past loop closures, without keeping them as active edges in the pose-chain. This filter-like mechanism is of critical importance and described further in Section II-B and demonstrated with an experiment in Section IV-E2.

Trajectory bending, which is at the core of COP-SLAM, is described first in general terms of Lie groups in the next section. A Lie group can be seen as a Riemannian manifold $\mathcal{M}$ together with a product structure $\star$ that acts on points of this Riemannian manifold (for more background on this, see [39]–[41]). In Section III, we provide detailed algorithms for specific Lie groups that are relevant to robotics. In Section II-C, we show that despite the single-loop simplification made by COP-SLAM, it is guaranteed to provide a solution that provably closes loops in linear time complexity in the number of edges for pose-chains in which nodes and edges can be modeled as elements of a Lie group $\mathcal{M},\star$ for which exponential and logarithmic maps can be computed such that $\mathrm{M}=\exp(\log(\mathrm{M}))$ for all $\mathrm{M}\in\mathcal{M}$. In Section II-D, we furthermore show that when the Lie group's logarithmic map is related to a bi-invariant metric, i.e., $\|\log(\mathrm{M})\| = \|\log(\mathrm{C}\star\mathrm{M}\star\mathrm{C}^{-1})\|,\ \mathrm{C},\mathrm{M}\in\mathcal{M}$ and when uncertainties of edges are isotropic, then COP-SLAM provides optimal solutions for pose-chains in which loops do not interact with each other.

Algorithm 1 COP-SLAM

while running **do**
  Get edge from SLAM front-end
  if edge is a *loop-closing edge* **then**
    • Add loop-closing edge to end of pose-chain.
    • Close its loop using trajectory bending (see Section II-A) and transfer its statistical information to the pose-chain (see Section II-B). For details on specific solutions spaces SE(3) and SIM(3), see Algorithms 2 and 3.
    • Deactivate loop-closing edge.
  else
    • Add *successive edge* to the end of the pose-chain and compute new last absolute pose.
  end if
end while

A. Trajectory Bending

The task of trajectory bending on general Lie groups is to optimally update a chain of relative transformations (relative poses), such that the last absolute transformation (absolute pose) is equal to a desired transformation. In this study, the desired transformation is related to a loop detection, but it can also be related to GPS, AHRS, or any other absolute pose sensor. The conceptual working of closed-form trajectory bending is visualized in Fig. 3. Prior to describing it in detail, we need to introduce the following notations.

Let $\mathrm{M}_1,\mathrm{M}_2,\ldots,\mathrm{M}_n$ be the set of *relative transformations*, which are all elements of the Lie group $\mathcal{M},\star$. The elements of the set are strictly ordered in time, and their subscripts denote their time steps. Each transformation also comes with its own uncertainty measure, which are modeled with zero-mean isotropic Gaussians with variances $\sigma_1^2,\sigma_2^2,\ldots,\sigma_n^2$. Each Gaussian is locally expressed in the tangent space of the Riemannian manifold $\mathcal{M}$ associated with the Lie group at the position of its respective M_i.

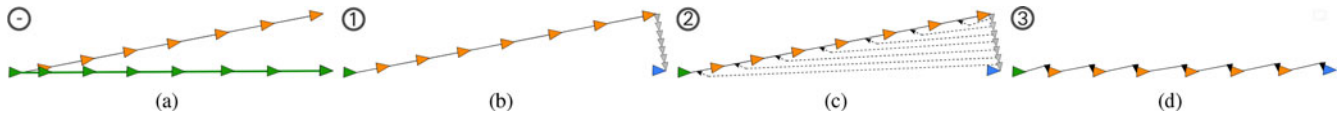

Fig. 3. Conceptual illustration of closed-form trajectory bending. In (a), the absolute poses of the ground-truth trajectory are depicted by the green triangles and the ground-truth relative pose-displacements by the green edges. Estimates for the absolute poses of the trajectory are depicted by the orange triangles and estimates for the relative pose-displacements by black edges. At a certain point in time, the system obtains more accurate information on its last absolute pose, i.e., the blue triangle in (b). The first step is to find relative pose updates, shown by the small light gray triangles in (b), which bring the last absolute pose onto the desired last pose, i.e., the blue triangle. These updates are called the *local updates*. The second step is to compute transformations, which distribute these local updates over the trajectory (c). The result of these transformations are the *distributed updates*, which are depicted in (d) by the small black triangles. In (d), an improved trajectory is computed using the relative pose-displacements together with their distributed updates. The result is that the trajectory ends in the desired absolute pose.

An initial estimate for the *absolute transformation* A_n at time step n can be computed with

$$\mathrm{A}_n = \prod_{i=1}^{n} \mathrm{M}_i = \mathrm{M}_1 \star \mathrm{M}_2 \star \mathrm{M}_3 \cdots \star \mathrm{M}_n \ . \tag{2}$$

The task of trajectory bending can now be formalized mathematically with

$$\mathrm{D}_n = \prod_{i=1}^{n} (\mathrm{M}_i \star \mathrm{U}_i) \tag{3}$$

i.e., each relative transformation M_i is improved with its own update U_i such that the last absolute transformation A_n becomes equal to a certain desired transformation D_n. The updates $\mathrm{U}_1, \mathrm{U}_2, \ldots, \mathrm{U}_n$ are called the *distributed updates*, and the desired transformation D_n puts a constraint on the last absolute transformation A_n, which is enforced exactly.

The absolute transformation (obtained from, e.g., loop detection) is denoted with L_n, and in all practical circumstances, this transformation is subject to uncertainty, which is modeled with σ_{L_n}. Because of this uncertainty, the desired transformation D_n, which is enforced exactly, cannot be L_n but must be an optimal combination of L_n and A_n. When the uncertainty in each relative transformation is isotropic and is Gaussian, and when the Lie group's logarithmic map is related to a bi-invariant Riemannian distance metric over the Riemannian manifold associated with the Lie group, then the uncertainty σ_{A_n} in A_n can be computed with

$$\sigma^2_{\mathrm{A}_n} = \sum_{i=1}^{n} \sigma_i^2. \tag{4}$$

The optimal fusion of A_n and L_n is then

$$\mathrm{D}_n = \mathrm{A}_n \star \exp\left(\frac{\sigma^2_{\mathrm{A}_n}}{\sigma^2_{\mathrm{A}_n} + \sigma^2_{\mathrm{L}_n}} \log\left(\mathrm{A}_n^{-1} \star \mathrm{L}_n\right) \right). \tag{5}$$

The transformation D_n lies on the geodesic from A_n to L_n such that the ratio of the geodesic distances between A_n to D_n and D_n to L_n is equal to $\sigma^2_{\mathrm{A}_n}/\sigma^2_{\mathrm{L}_n}$. The uncertainty in the fused result D_n is

$$\sigma^2_{\mathrm{D}_n} = \frac{1}{1/\sigma^2_{\mathrm{A}_n} + 1/\sigma^2_{\mathrm{L}_n}}. \tag{6}$$

The fusion in (5) does not need to be performed before applying trajectory bending but is taken care of inside trajectory bending itself, as is explained next.

1) Closed-Form Solution: The concept behind the closed-form solution to trajectory bending is to optimally transfer the error at the loop-closing edge over all successive edges. The error at the loop-closing edge is the transformation between the last transformation A_n and the desired absolute transformation D_n, which is provided with $\mathrm{A}_n^{-1} \star \mathrm{D}_n$. The successive edges are the relative transformations $\mathrm{M}_1, \mathrm{M}_2, \ldots, \mathrm{M}_n$, and the goal is to optimally distribute the error $\mathrm{A}_n^{-1} \star \mathrm{D}_n$ over these relative transformations by computing relative updates $\mathrm{U}_1, \mathrm{U}_2, \ldots, \mathrm{U}_n$. This is performed by a three-step closed-form process.

The first step [see Fig. 3(b)] is to compute the n *local updates* $\hat{\mathrm{U}}_1, \hat{\mathrm{U}}_2, \ldots, \hat{\mathrm{U}}_n$ such that

$$\mathrm{D}_n = \mathrm{A}_n \star \prod_{i=1}^{n} \hat{\mathrm{U}}_i \tag{7}$$

i.e., when putting all local updates behind the last absolute transformation A_n, the result is equal to the desired transformation D_n. Each local update can be computed from the last absolute pose A_n and the absolute pose L_n (obtained from, e.g., loop detection) using the relative interpolation function

$$\hat{\mathrm{U}}_i = \left[\mathfrak{I}\left(\sum_{j=1}^{i-1} w_j \right) \right]^{-1} \mathfrak{I}\left(\sum_{j=1}^{i} w_j \right) \tag{8}$$

where

$$\mathfrak{I}(\alpha) = \exp(\alpha \log(\mathrm{A}_n^{-1} \star \mathrm{L}_n)). \tag{9}$$

The normalized weights $w_1, w_2, \ldots, w_n$ are computed from the variances with

$$w_j = \frac{\sigma_j^2}{\sigma^2_{\mathrm{L}_n} + \sigma^2_{\mathrm{A}_n}} \tag{10}$$

and determine how much of the update is distributed to a particular relative transformation in the chain. The more uncertain a transformation is, the more it will be improved relatively to the other relative transformations. In Section II-D1, it is proven that when using these particular weights, trajectory bending obtains a maximum likelihood solution.

This first step automatically performs the optimal fusion between A_n and L_n to come to D_n. To see this, observe that for the last term in (7), we have

$$\prod_{i=1}^{n} \hat{\mathrm{U}}_i = \prod_{i=1}^{n} \left(\left[\mathfrak{I}\left(\sum_{j=1}^{i-1} w_j \right) \right]^{-1} \mathfrak{I}\left(\sum_{j=1}^{i} w_j \right) \right)$$

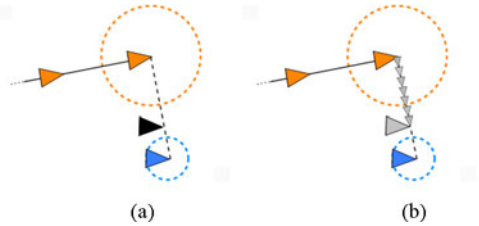

Fig. 4. (a) External and (b) internal fusion mechanisms for trajectory bending. The uncertainty in the last absolute pose is depicted by the orange circle and the uncertainty in the loop-closing pose by the blue circle. The minimal length geodesic between the last absolute pose and the loop-closing pose is depicted by the dashed black line. The optimal combination of the last absolute pose and the loop-closing pose lies on this geodesic and is shown by the black triangle in (a). It is called the desired pose and can be computed externally in one step by using (5). In (b), the local updates of trajectory bending, obtained by interpolating with (8) over the geodesic, end at the large gray triangle, which has the same position as the desired pose.

$$= \mathfrak{I}\left(\sum_{j=1}^{n} w_j\right) = \mathfrak{I}\left(\frac{\sum_{j=1}^{n}\sigma_j^2}{\sigma_{\mathrm{L}_n}^2+\sigma_{\mathrm{A}_n}^2}\right)$$

$$= \mathfrak{I}\left(\frac{\sigma_{\mathrm{A}_n}^2}{\sigma_{\mathrm{L}_n}^2+\sigma_{\mathrm{A}_n}^2}\right)$$

$$= \exp\left(\frac{\sigma_{\mathrm{A}_n}^2}{\sigma_{\mathrm{L}_n}^2+\sigma_{\mathrm{A}_n}^2}\log\left(\mathrm{A}_n^{-1}\star\mathrm{L}_n\right)\right). \quad (11)$$

If we now use the last equality to rewrite (7), we obtain

$$\mathrm{D}_n = \mathrm{A}_n \star \exp\left(\frac{\sigma_{\mathrm{A}_n}^2}{\sigma_{\mathrm{A}_n}^2+\sigma_{\mathrm{L}_n}^2}\log\left(\mathrm{A}_n^{-1}\star\mathrm{L}_n\right)\right). \quad (12)$$

It is the optimal fusion formula of (5). What happens is that the relative interpolation function moves over the geodesic from A_n to L_n in n steps until the point of optimal fusion D_n is reached; see Fig. 4 for a conceptual illustration of this process.

The second step [see Fig. 3(c)] is to distribute the local updates over all relative transformations. We thus seek a set of *distributed updates* $\mathrm{U}_1, \mathrm{U}_2, \ldots, \mathrm{U}_n$ such that

$$\mathrm{D}_n = \mathrm{A}_n \star \prod_{i=1}^{n}\hat{\mathrm{U}}_i = \prod_{i=1}^{n}(\mathrm{M}_i \star \mathrm{U}_i). \quad (13)$$

Instead of putting all the updates behind the last absolute transformation A_n, each relative transformation M_i is succeeded by its own distributed update U_i. The relation between each local update $\hat{\mathrm{U}}_i$ and its distributed update U_i is given by the map

$$\mathfrak{T}(\hat{\mathrm{U}}_i) = \mathrm{A}_i^{-1}\star\mathrm{D}_n\star\hat{\mathrm{U}}_i\star\mathrm{D}_n^{-1}\star\mathrm{A}_i = \mathrm{U}_i. \quad (14)$$

A proof for this map is provided in Section II-C. The interesting property is that the map $\mathfrak{T}$ for each $\hat{\mathrm{U}}_i$ only depends on the original absolute pose A_i and the desired absolute pose D_n. The solution to trajectory bending can, therefore, be obtained in closed form and has computational complexity $\mathrm{O}(n)$. As the mapping can be computed independently for each $\hat{\mathrm{U}}_1, \ldots, \hat{\mathrm{U}}_n$, the memory requirements are constant for volatile memory (e.g., RAM) and linear in n for nonvolatile memory (e.g., a hard disk or flash memory). This allows for straightforward distributed (out-of-core) processing for big pose-chains. These aspects make the trajectory bending algorithm significantly more efficient than alternative iterative approaches.

The third and final step [see Fig. 3(d)] is to update the relative transformations and their uncertainties as well as to compute the new absolute transformations. Updating the relative and absolute transformations can be done with $\mathrm{M}_i \star \mathrm{U}_i \rightarrow \mathrm{M}_i$ and then using (2). Updating the uncertainties of the relative transformations, as modeled by the variances $\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2$, is more involved and explained in the next section.

B. Multiple Loops and Filter-Like Behavior

Updating the variances $\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2$ is crucial when applying trajectory bending in the context of COP-SLAM. It allows utilizing statistical information of previous loop-closing edges, without the need to keep them as active edges in the pose-chain.

The variances express how accurate a certain relative transformation is expected to be with respect to the other relative transformations. When closing a loop, the desired absolute transformation adds important information making all the relative transformations within the loop more accurate with respect to relative transformations that were not inside the loop. When a new loop is detected that contains relative transformations that were part of previous loops and relative transformations that were not part of previous loops, we want to focus bending more on the relative transformations that were not part of the previous loop, as these are less accurate with respect to relative transformations that were part of previous loops. In a sense, we do not want to destroy previously closed loops, when closing new loops. What we, therefore, need is a mechanism to optimally compute new values for the variances of relative transformations involved in loop closing, such that their increase in accuracy is accounted for when closing future loops.

This mechanism is obtained by considering that before a loop is closed, the uncertainty of the last absolute transformation is obtained by summing all relative transformation uncertainties with (4). After loop closure, the uncertainty in the updated last transformation is provided by (6). The goal is to update all relative transformation uncertainties $\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2$ such that when they are summed up in (4), the result is equal to the new uncertainty of the last transformation provided by (6). Put in mathematical terms, we seek a scalar β such that

$$\sum_{i=1}^{n}\beta\sigma_i^2 = \frac{1}{1/\sigma_{\mathrm{A}_n}^2+1/\sigma_{\mathrm{L}_n}^2}. \quad (15)$$

After some straightforward manipulation, we find that β is provided by

$$\beta = \frac{1}{1+\sigma_{\mathrm{A}_n}^2/\sigma_{\mathrm{L}_n}^2} \quad (16)$$

and the variances can be updated with $\beta\sigma_i^2 \rightarrow \sigma_i^2$.

At the conceptual level, this mechanism has similarities with a Kalman filter. Adding relative transformations to the pose-chain can be seen as a *prediction step* and increases the uncertainty of the final absolute transformations. When detecting a loop,

the final absolute transformation is optimally fused with the loop-closing transformation [see (5)], much like as is done in the *update step* of a Kalman filter, resulting in the desired final absolute transformation. Next, all relative transformations in the loop are updated in accordance with the desired final absolute transformation with trajectory bending. Similar to the *update step* in a Kalman filter, the uncertainty in the desired final absolute transformation is also recomputed ([see (6)]. In COP-SLAM, this uncertainty is then back-propagated to the uncertainty of all relative transformation in the loop, by updating their variances [see (16)]. This ensures that the statistical information of the current loop closure is represented for in the pose-chain, when closing future loops, although it will no longer be modeled as an active edge in the pose-chain. The critical importance of this filter-like behavior is demonstrated with an experiment in Section IV-E2.

C. Convergence

Here, we prove that COP-SLAM converges in a single step to a solution that closes each new loop exactly (i.e., to within machine precision). The variance updating strategy of Section II-A assures that previously closed loops are respected in a manner that is as optimal as can be without explicitly modeling past loop-closing edges. The only prerequisite is that local updates can be computed with the relative interpolation function (9), which requires that exponential and logarithmic maps can be computed such that $\mathrm{M} = \exp(\log(\mathrm{M}))$ for all $\mathrm{M} \in \mathcal{M}$. Given this prerequisite, we only need to prove that the relation between the local updates and the distributed updates is provided by the map $\mathfrak{T}$ of (14). This is captured mathematically by the following theorem.

Theorem 1: A map from the local updates $\hat{\mathrm{U}}_1 \cdots \hat{\mathrm{U}}_n$ to the distributed updates $\mathrm{U}_1 \cdots \mathrm{U}_n$ obeying the loop-closing constraint (3) is provided by

$$\begin{aligned} \mathfrak{T}(\hat{\mathrm{U}}_i) &= \mathrm{A}_i^{-1}\mathrm{D}_n\hat{\mathrm{U}}_i\mathrm{D}_n^{-1}\mathrm{A}_i \\ &= \left(\prod_{j=1}^{i}\mathrm{M}_j\right)^{-1}\mathrm{D}_n\hat{\mathrm{U}}_i\mathrm{D}_n^{-1}\left(\prod_{j=1}^{i}\mathrm{M}_j\right). \\ &= \mathrm{U}_i. \end{aligned} \tag{17}$$

The details of the proof for Theorem 1 are in supplementary material [42, Sec. 3], which can be skipped at first reading. The proof is based on induction and starts with the *proposition*

$$P(k): \quad \prod_{j=1}^{n-k}\mathrm{M}_j \prod_{j=n-k+1}^{n}(\mathrm{M}_j\mathfrak{T}(\hat{\mathrm{U}}_j)) \prod_{j=1}^{n-k}\hat{\mathrm{U}}_j = \mathrm{D}_n. \tag{18}$$

For $k = 0$, the proposition conceptually expresses that when putting all local updates behind the last absolute transformation, the result is equal to the desired transformation D_n. This is the initial step of trajectory bending, depicted in Fig. 3(b), and the validity of $P(0)$ is guaranteed by using the relative interpolation function (8). For $k = n$, the proposition conceptually expresses that when all updates are distributed over the trajectory, it ends in the desired absolute transformation. This is the end goal of the trajectory bending algorithm and depicted in Fig. 3(d).

In summary, to prove Theorem 1, the proposition $P(k)$ needs to be valid for all $1 \leq k \leq n$. The *basis* for induction is then $P(1)$, whose validity is proven by algebraic manipulation. The next step is to prove the *inductive step*, i.e., the correctness of $P(k+1)$ when assuming that the *inductive hypothesis* $P(k)$ holds. This is also performed using algebraic manipulation, and the proof then follows from induction.

D. Optimality

As COP-SLAM uses trajectory bending to close loops, its optimality conditions are mainly determined by the optimality conditions of trajectory bending, which are, therefore, described first in Section II-D1. Those of COP-SLAM itself are provided in Section II-D2.

1) Trajectory Bending: Trajectory bending can be seen as a constrained optimization task. It requires a solution that closes the loop (i.e., the constraint) such that the trajectory is bent minimally (i.e., the cost function). The constraint is given by (3) and is repeated here for completeness

$$\mathrm{D}_n = \prod_{i=1}^{n}(\mathrm{M}_i \star \mathrm{U}_i). \tag{19}$$

When uncertainties in relative transformations are isotropic, a maximum likelihood objective function for trajectory bending can be formulated as

$$f(\mathrm{U}_1, \ldots, \mathrm{U}_n) = \prod_{i=1}^{n}\frac{1}{\sqrt{2\pi\sigma_i^2}}e^{\frac{\|\log(\mathrm{U}_i)\|^2}{2\sigma_i^2}}. \tag{20}$$

It expresses that we are seeking those distributed updates, which are most likely given the uncertainties of the relative transformations. This likelihood function is expressed over the Riemannian manifold $\mathcal{M}$ related to the Lie group $\mathcal{M}, \star$, and $\|\log(\mathrm{U}_i)\|$ is the Euclidean metric computed on the vectorized Lie algebra element $\log(\mathrm{U}_i)$ (i.e., a tangent vector of the Riemannian manifold $\mathcal{M}$). We now prove the following theorem.

Theorem 2: Trajectory bending provides a maximum likelihood solution to (20) under the constraint (19) when the logarithmic map of the Lie group $\mathcal{M}, \star$ allows for a bi-invariant metric $\|\log(\mathrm{M})\| = \|\log(\mathrm{C} \star \mathrm{M} \star \mathrm{C}^{-1})\|,\ \mathrm{C}, \mathrm{M} \in \mathcal{M}$ by setting the interpolation weights to

$$w_i = \frac{\sigma_i^2}{\sigma_{\mathrm{L}_n}^2 + \sum_{i=1}^{n}\sigma_i^2}. \tag{21}$$

The details of the proof for Theorem 2 are in supplementary material [42, Sec. 4], which can be skipped at first reading. In summary, the proof consists of two parts. In the first part, we show that the constraint and the likelihood function, which are expressed in terms of distributed updates, can be expressed alternatively in terms of local updates. The key to this is the availability of a bi-invariant metric. In the second part, we show that the solution for trajectory bending presented in Section II-A provides a maximum likelihood solution for the local updates.

2) Closed-Form Online Pose-Chain Simultaneous Localization and Mapping: So far, we have discussed the optimality conditions of trajectory bending when applied to a single loop.

Here, we discuss the optimality conditions of COP-SLAM when applied to pose-chains that generally consist of several entangled loops. Even when assuming isotropic noise, obtaining maximum likelihood solutions for such pose-chains requires iterative nonlinear optimization. Because COP-SLAM's closed-form approach cannot propagate information of a loop-closure back past the start of the loop that is being closed, it can only provide maximum likelihood solutions if all loops in the pose-chains do not interact with each other, i.e., when the pose-chain is a sequence of multiple pure pose-chains. We, therefore, say that COP-SLAM generally provides *loop-wise* optimal solutions. The more entangled a pose-chain is (i.e., the more it looks like a general pose-graph), the bigger the expected difference in accuracy is between COP-SLAM's closed-form solution and that of iterative optimizers. This is investigated experimentally in Section IV-D3.

Another fundamental aspect is that in order for trajectory bending to be optimal and hence for COP-SLAM to be loop-wise optimal, the Lie group must allow for a bi-invariant metric. The most relevant Lie groups for robotics, i.e., the group of Euclidean motions and the group of similarity transforms, do, however, not allow for such a bi-invariant metric [43]. The optimality of iterative optimizers is not harmed by this because they can actively alter relative rotation or relative scale estimates to reduce absolute position errors. COP-SLAM cannot makes such tradeoffs actively, but in Section III, we show that COP-SLAM can do this passively by applying a (subspace) stratified trajectory bending approach. In Section IV, we investigate experimentally how close the accuracy of such stratified algorithms is to the accuracy of iterative optimizers that provide maximum likelihood solutions.

III. Solution Spaces and Their Optimality

In previous sections, we have discussed COP-SLAM in general terms of Lie groups. In this section, we provide stratified algorithms for the specific solution spaces of Euclidean motions SE(3) and that of similarity transforms SIM(3). We, furthermore, show that the accuracy of these stratified algorithms is between specific lower and upper bounds. All our conclusions for these solution spaces also apply to their 2-D counterparts SE(2) and SIM(2).

A. SE(3)

Let us first introduce the efficient notation $\mathrm{A} = [\mathrm{R}, \mathbf{t}]$, $\mathrm{R} \in \mathrm{SO}(3)$, $\mathbf{t} \in \mathbb{R}^3$ to express an element of SE(3). The Lie group product and the inverse of SE(3) can then be denoted with

$$\begin{aligned} \mathrm{A}_1 \star \mathrm{A}_2 &= [\mathrm{R}_1\mathrm{R}_2, \mathrm{R}_1\mathbf{t}_2 + \mathbf{t}_1] \\ \mathrm{A}^{-1} &= [\mathrm{R}^\top, -\mathrm{R}^\top\mathbf{t}]. \end{aligned} \tag{22}$$

As noted earlier, the Lie group SE(3) does not allow for a bi-invariant metric. Conceptually, this is because its subgroup SO(3) and its subgroup $\mathbb{R}^3$ are geometrically coupled such that SO(3) acts on $\mathbb{R}^3$. This prevents COP-SLAM from obtaining loop-wise optimal solutions, because it cannot actively make tradeoffs between rotational and translational errors. Both subgroups SO(3) and $\mathbb{R}^3$ do allow for a bi-invariant metric themselves and so does their direct product space $\mathrm{SO}(3) \times \mathbb{R}^3$ in which SO(3) does not act on $\mathbb{R}^3$. This can be utilized to derive a stratified algorithm together with upper and lower bounds on its accuracy.

Deriving the lower bound is based on decoupling SO(3) and $\mathbb{R}^3$ when integrating the relative poses into absolute poses. At the time of estimating the relative poses, this decoupling is not possible as a relative translation of time t is expressed with respect to a basis with the absolute orientation of time $t-1$. However, at the moment of loop closure, this decoupling becomes possible. We first compute all absolute poses in the chain from the relative poses by using the Lie group structure of SE(3). Once all absolute positions and orientations are known, the relative pose-displacements can alternatively be expressed as elements of the Lie group $\mathrm{SO}(3) \times \mathbb{R}^3$. These decoupled relative pose-displacements can be obtained from the absolute poses with

$$\mathrm{M}_n = \mathrm{A}_{t-1}^{-1} * \mathrm{A}_t \tag{23}$$

but now using the Lie group product and inverse of $\mathrm{SO}(3) \times \mathbb{R}^3$, which are

$$\begin{aligned} \mathrm{A}_1 * \mathrm{A}_2 &= [\mathrm{R}_1\mathrm{R}_2, \mathbf{t}_1 + \mathbf{t}_2] \\ \mathrm{A}^{-1} &= [\mathrm{R}^\top, -\mathbf{t}]. \end{aligned} \tag{24}$$

The exponential and logarithmic maps of $\mathrm{SO}(3) \times \mathbb{R}^3$ are

$$\begin{aligned} \log(\mathrm{M}) &= [\log(\mathrm{R}), \mathbf{t}] = [\mathfrak{r}, \mathbf{t}] = \mathfrak{m} \\ \exp(\mathfrak{m}) &= [\exp(\mathfrak{r}), \mathbf{t}] = [\mathrm{R}, \mathbf{t}] = \mathrm{M} \end{aligned} \tag{25}$$

and its bi-invariant metric is

$$\| \log(\mathrm{M}) \| = \sqrt{\mathfrak{r}_x^2 + \mathfrak{r}_y^2 + \mathfrak{r}_z^2 + \mathbf{t}_x^2 + \mathbf{t}_y^2 + \mathbf{t}_z^2} \tag{26}$$

which can be generalized in the usual way. The functional expressions for the exponential and logarithm maps for SO(3) can be found in [44, p. 42].

By using the Lie group $\mathrm{SO}(3) \times \mathbb{R}^3$, COP-SLAM closes the loop exactly in SO(3) and in $\mathbb{R}^3$ such that it is loop-wise optimal with respect to the metric (26). This is what we call a monolithic approach, as COP-SLAM acts simultaneously on all subgroups of $\mathrm{SO}(3) \times \mathbb{R}^3$. We can improve on the accuracy of this monolithic approach by using stratification. In our stratified approach, provided in Algorithm 2, we first apply trajectory bending solely to the subgroup of rotations SO(3). This will close the loop in the rotational subspace and will improve all relative rotations on average. Then, we reintegrate the trajectory utilizing SE(3) to obtain improved absolute positions given the improved relative rotations. Finally, we perform trajectory bending to the translational subgroup $\mathbb{R}^3$ to nullify any absolute position residual errors in the loop-constraint. The end results is that again the loop is closed in both the rotation and translation subspaces, but now, we have (passively) used improvements in relative rotations to improve absolute positions. This stratified approach is, therefore, theoretically more accurate than the monolithic approach of using $\mathrm{SO}(3) \times \mathbb{R}^3$ but also theoretically less accurate than an iterative optimizer using SE(3). This gives clear lower

and upper bounds on the accuracy of stratified COP-SLAM when applied to pose-chains consisting of Euclidean motions.

Note that using the Lie group SE(3) and its exponential and logarithm maps within monolithic COP-SLAM will generally lead to even less accurate results than the monolithic approach using SO(3) $\times \mathbb{R}^3$. This is because the one-parameter subgroups (used for computing the local updates) of SE(3) through its subspace $\mathbb{R}^3$ are not straight lines and, therefore, not minimizing geodesics in $\mathbb{R}^3$. Only iterative optimizers can truly overcome this by actively seeking tradeoffs between rotational and translational errors.

Algorithm 2 Stratified trajectory bending in SE(3) To be used inside algorithm 1

Restrict computations to the Lie group of rotations SO(3).
R.1) Compute the local updates with (9).
R.2) Compute the distributed updates with (14).
R.3) Update all relative orientations given the distributed updates and multiply each σ^2 of each relative orientation with the factor β of (16).
Recompute all absolute orientations and absolute positions given the updated relative orientations using SE(3).
Restrict computations to the Lie group of translations $\mathbb{R}^3$.
T.1) Compute the local updates with (9).
T.2) Compute the distributed updates with (14).
T.3) Update all relative translations given the distributed updates and multiply each σ^2 of each relative translation with the factor β of (16).
Recompute all absolute positions given the updated relative translations.

B. SIM(3)

We now introduce a new solution space that was not considered before in the context of closed-form trajectory bending or in the context of COP-SLAM. It is the Lie group of similarity transforms SIM(3), i.e., Euclidean motions with uniform scaling. This group has been considered earlier in the context of iterative optimization methods [25] and allows closing the loop under scale-drift, which is a common nuisance in monocular odometry. The Lie group of similarity transforms does also not allow for a bi-invariant metric but, similar to SE(3), a stratified trajectory bending algorithm can be derived.

Let us introduce an efficient notation $\mathrm{A} = [\mathrm{R}, \mathbf{t}, s]$, $\mathrm{R} \in$ SO(3), $\mathbf{t} \in \mathbb{R}^3$ and $s \in \mathbb{R}^\times/0$ to express an element of SIM(3) (here, $\mathbb{R}^\times/0$ denotes the group of real numbers modulo 0 under multiplication). Its group operator and inverse are

$$\begin{aligned} \mathrm{A}_1 \circ \mathrm{A}_2 &= [\mathrm{R}_1\mathrm{R}_2, s_1\mathrm{R}_1\mathbf{t}_2 + \mathbf{1}_2, s_1 s_2] \\ \mathrm{A}^{-1} &= \left[\mathrm{R}^\top, -\mathrm{R}^\top\frac{1}{s}\mathbf{t}, \frac{1}{s}\right]. \end{aligned} \tag{27}$$

The stratified algorithm for SIM(3) then follows in a similar fashion as that of SE(3). First, we decouple all subgroups of SIM(3) by introducing the direct product space SO(3) $\times \mathbb{R}^3 \times \mathbb{R}^\times/0$, which has the following group operator and inverse:

$$\mathrm{A}_1 \diamond \mathrm{A}_2 = [\mathrm{R}_1\mathrm{R}_2, \mathbf{t}_1 + \mathbf{t}_2, s_1 s_2]$$

$$\mathrm{A}^{-1} = [\mathrm{R}^\top, -\mathbf{t}, \frac{1}{s}] \tag{28}$$

and its logarithmic and exponential mappings are

$$\begin{aligned} \log(\mathrm{M}) &= [\log(\mathrm{R}), \mathbf{t}, \log(s)] = [\mathfrak{r}, \mathbf{t}, \mathfrak{s}] = \mathfrak{m} \\ \exp(\mathfrak{m}) &= [\exp(\mathfrak{r}), \mathbf{t}, \exp(\mathfrak{s})] = [\mathrm{R}, \mathbf{t}, s] = \mathrm{M} \end{aligned} \tag{29}$$

where $\log(s)$ and $\exp(\mathfrak{s})$ are the usual (base 10) logarithm and exponent for natural numbers. The bi-invariant metric of SO(3) $\times \mathbb{R}^3 \times \mathbb{R}^\times/0$ is

$$\|\log(\mathrm{M})\| = \sqrt{\mathfrak{r}_x^2 + \mathfrak{r}_y^2 + \mathfrak{r}_z^2 + \mathbf{t}_x^2 + \mathbf{t}_y^2 + \mathbf{t}_z^2 + \mathfrak{s}^2} \tag{30}$$

which again can be generalized in the usual way.

The stratified algorithm for similarity transforms is provided in Algorithm 3. It allows utilizing improvements in relative scales and in relative rotations to improve absolute positions. For similar reasons as for the stratified algorithm for SE(3), this stratified algorithms is theoretically more accurate than monolithic trajectory bending using SIM(3) or monolithic trajectory bending using SO(3) $\times \mathbb{R}^3 \times \mathbb{R}^\times/0$ but less accurate than an iterative optimizer using SIM(3).

Algorithm 3 Stratified trajectory bending in SIM(3) To be used inside algorithm 1

Restrict computations to the Lie group of scaling $\mathbb{R}^\times/0$.
S.1) Compute the local updates with (9).
S.2) Compute the distributed updates with (14).
S.3) Update all relative scale factors given the distributed updates and multiply each σ^2 of each relative scaling with the factor β of (16).
Restrict computations to the Lie group of rotations SO(3).
R.1) Compute the local updates with (9).
R.2) Compute the distributed updates with (14).
R.3) Update all relative orientations given the distributed updates and multiply each σ^2 of each relative orientation with the factor β of (16).
Recompute all absolute orientations, absolute scale factors and absolute positions given the updated relative orientations an updated relative scale factors using SIM(3).
Restrict computations to the Lie group of translations $\mathbb{R}^3$.
T.1) Compute the local updates with (9).
T.2) Compute the distributed updates with (14).
T.3) Update all relative translations given the distributed updates and multiply each σ^2 of each relative translation with the factor β of (16).
Recompute all absolute positions given the updated relative translations.

In Section IV, we use this novel stratified algorithm for similarity transforms together with landmark position updating as a closed-form initialization strategy for sequential bundle adjustment.

C. Note on Efficiency

At first, it may seem that a stratified approach requires more computation than a monolithic approach, as we are applying

trajectory bending twice. The opposite is true, it is more efficient. For example, for Euclidean motions, the number of floating point operations required when dealing with rotations and translation separately is less than when dealing with them within SE(3). Multiplying two elements of SE(3) will take 36 multiplications and 27 additions. Multiplying two element of SO(3) takes 27 multiplications and 18 additions and adding two translations takes three additions. Therefore, there is a total of only 27 multiplications and only 21 additions, when dealing with rotations and translations separately. A similar reduction is observed for computing the inverse. These reductions in floating point operations compensate for the additional steps of the stratified algorithm.

IV. Evaluation

The main body of our experiments specifically target online 3-D pose-chain SLAM, where the pose-chain is obtained by using accurate visual odometry and reliable appearance-based loop detection. Within this scope, we investigate the performance differences between monolithic COP-SLAM, stratified COP-SLAM, and an iterative method that provides maximum likelihood solutions. From our theoretical analysis of COP-SLAM, it is already clear that it generally cannot obtain similar accuracy as iterative methods. The accuracy of iterative methods is, however, partially determined by the density and the topology of the pose-graphs to which they are applied, i.e., the more (loop-closing) edges, the more accurate its solution will be. It is, therefore, relevant to evaluate how much benefits remain of iterative methods when applying them to pose-chains, which are extremely sparse by nature, and to compare this to the performance of COP-SLAM, which is specifically designed to optimize pose-chains. We do this on basis of simulated data as well as extensive real-world binocular data and monocular data.

In recent work [9], several iterative methods were compared against the G^{2}O back-end optimizer. It was shown that G^{2}O provides either better or comparable accuracy and efficiency than alternative optimization methods. Therefore, here, we only compare to G^{2}O. During initial experiments, we have observed that the Gauss–Newton method (which does not perform a *line search*) with the CHOLMOD solver of G^{2}O outperforms other solvers in accuracy and efficiency on our datasets. Therefore, we only report the performance using this optimizer and solver. The original online version of G^{2}O runs the optimizer every time a certain number of edges has been added. For pose-chains, this is an inefficient approach, as *successive edges* do not add information to the chain that is of value to past nodes. We have, therefore, modified the original code of G^{2}O such that it only runs the optimizer when a *loop-closing edge* is added to the chain. This allows G^{2}O to save a huge amount of unnecessary computations.

All our experiments are performed on a laptop using a single core of a Core 2 Duo T9400 CPU (2.53 GHz) accompanied by 4 GB of SDRAM. Both COP-SLAM and G^{2}O run equally optimized C/C++ code based on SSE instructions. Our COP-SLAM program along with the pose-chains used for our experiments are made publicly available in [32]. Before discussing the experimental results, we first provide important details on the used performance metrics, our SLAM front-ends, and the datasets.

A. Performance Metrics

To provide detailed insight into the performance of COP-SLAM, we use several metrics during our evaluations. The focus of our experiments is on SLAM systems that use accurate vision-based front-ends. Because of this, the error of successive edges (relative poses) is extremely low, typically in the order of millimeters and millidegrees, making it virtually impossible to directly measure improvements at the granularity of successive edges, e.g., on basis of (RTK-)GPS ground truth. The errors of successive edges will, however, accumulate in the nodes (absolute poses), resulting in significant errors, which can be measured adequately using (RTK-)GPS ground truth. Therefore, our performance metrics are based on errors in absolute positions and on errors in absolute orientations averaged over the number of nodes. These metrics are reported as absolute values and as percentages of the initial error of the visual-SLAM front-end. The latter allows for more direct comparison of errors between different datasets.

To prevent overconfident error measurements, due to estimated absolute positions coincidentally being close to ground-truth positions, all our error measurements are based on time stamps. Hence, an estimated pose is only compared with the ground-truth pose, having the same time stamp. For this, we linearly interpolate the GPS trajectory, such that for each (key-)frame, there is an interpolated GPS ground-truth measurement. For certain datasets, the global orientation of the first pose is not known with sufficient accuracy. Therefore, before error computation, we automatically align the trajectory with the ground truth. To again prevent overconfident error measurements, we only use the first 50% of the trajectory when aligning to the ground truth.

In accordance with [45], we also use the *Chi square error* as a performance metric for specific experiments. This metric measures the residual of the SLAM cost function after optimization. Comparing SLAM methods on basis of this metric is only truly useful if both optimize exactly the same cost function. In our setting, this is not the case. COP-SLAM minimizes the error between the final absolute pose and the desired final absolute pose to zero (within machine precision). It thereby implicitly optimizes an approximation of the cost function of G^{2}O. The two cost functions are, therefore, not directly comparable. To overcome this, we load the solution of COP-SLAM into G^{2}O and measure the initial Chi square error, i.e., the error before G^{2}O optimization, and compare this to the final Chi square error of G^{2}O's solution. This gives an indication of how well COP-SLAM implicitly optimizes the nonlinear maximum likelihood cost function of G^{2}O.

B. Simultaneous Localization and Mapping Front-End

To show the versatility of COP-SLAM, we evaluate it in combination with different state-of-the-art visual odemetry methods in the SLAM front-end. The first binocular odometer that we use is a key-frame method that is similar to the one used in

(a) (b) (c) (d)

Fig. 5. Example images from the three binocular Pittsburg datasets, used for our experiments.

[23]. It provides a robust maximum likelihood solution by minimizing reprojection errors over two stereo frames. It outputs relative poses $\mathrm{M}_{i-1,i}$ (i.e., successive edges) accompanied by anisotropic 6×6 covariance matrices $\Sigma_{i-1,i}$, which are obtained by linear error propagation. Conceptually, these covariance matrices represent the effect of the uncertainty of all inlier image features on the relative pose estimates. The second stereo odometer is LIBVISO2 [38]. This RANSAC-based method does not provide covariance matrices, but does provide the number and the percentage of image feature inliers, which can also be used as a measure for the reliability of the odometry estimate. The monocular odometer that we use is a key-frame sliding-window method similar to the one used in [46], but now only optimizing for a single camera view.

In the front-end, state-of-the-art loop detection is done by feeding every (key-)frame into RTAB-MAP[27]. Several loop-verification strategies are used inside RTAB-MAP (e.g., multiframe consistency check, multiframe winner-margin check), and once it detects a loop, it is fed to an external RANSAC strategy in order to robustly obtain the loop-closing edge. If the number of landmark inliers obtained by the RANSAC strategy is less than 50, the detected loop is rejected; otherwise, the loop-closing edge ${}^{j}\mathrm{L}_{k,m}$ and its uncertainty ${}^{j}\Sigma_{k,m}$ are computed on all inlier image features. This whole procedure allows for extremely robust and accurate loop detection at the expense of detecting many loops and at the expense of detecting loops in repetitive environments. In other words, with respect to loop detection, we explicitly prefer *precision* over *recall*.

Please note that both COP-SLAM and G^{2}O use every loop that is detected by the SLAM front-end. There is no marginalization, loop selection, or sparsification being used, and the input to COP-SLAM and to G^{2}O is exactly the same. Consequently, G^{2}O uses all information that is provided by the state-of-the-art visual SLAM front-ends, and only COP-SLAM internally approximates the covariances matrices Σ of each edge by two scalars, i.e., one for the translation σ_{t}^2 and one for the rotation σ_{r}^2.

C. Datasets

To demonstrate the applicability of COP-SLAM, we use challenging publicly available visual datasets as well as specifically recorded datasets, of which representative images are shown in Fig. 5. Our experiments target online 3-D SLAM applications and large-scale outdoor environments in which frequent loop detections are not desired or not possible. Although these types of applications do not cover the complete scope of SLAM, they are nevertheless very relevant. We, however, find that many standard SLAM benchmark datasets do not target the same type of applications and environments. A notable exception is the KITTI benchmark dataset [35], but even its visual odometry datasets are not as challenging as the ones we specifically recorded for our experiments. To illustrate this, we show properties and statistics of our datasets (Pittsburgh A,B,C), that of the longest KITTI datasets with loops, and also that of the frequently used New College dataset [47] in Table I. These are all stereo vision datasets.

The key information in Table I is the length of each dataset and the percentage of overlap in the trajectory. This percentage is representative for the portion of the trajectory where loop detection is possible. It is determined by measuring the number of poses, which have nonconsecutive poses with a somewhat similar position and orientation. It can clearly be observed that our datasets (Pittsburgh A,B,C) are longer and have less opportunities to close loops than the other datasets. Furthermore, only our datasets and the KITTI datasets are accompanied by ground truth that is of sufficient accuracy for quantitative comparisons. Nevertheless, we also process the New College dataset to qualitatively show that COP-SLAM can also be used for denser pose-graphs.

To, furthermore, show the extensibility of COP-SLAM, we apply it to a monocular dataset, which besides drift in position and orientation also suffers from drift in scale. For this experiment, we use the novel SIM(3) solution space on a 800-m-long trajectory recorded by a monocular camera (640×480 gray scale, 6-mm focal length, 30 frames/s) mounted on top of a vehicle. Although this dataset potentially allows using other methods to prevent scale drift (e.g., by utilizing the fixed camera height above the ground plane or measuring wheel rotations), we use it because it has accurate ground truth based on differential GPS (3-cm standard deviation). This ground truth is important for our quantitative comparisons and is more difficult to obtain for, e.g., an (indoor) hand-held monocular dataset in which scale drift cannot be prevented by alternative methods.

We also use simulated data for precise comparison of the accuracy of COP-SLAM with that of G^{2}O under specific circumstances. For these simulated datasets, we mimic the statistical error characteristics of binocular front-ends. We use ten different datasets, and for each dataset, 100 different pose-chains are randomly generated, resulting in a total of 1000 unique experiments. Fig. 6 shows typical pose-chains of our datasets. For the Loop and Flower datasets [see Fig. 6(a) and (b)], which encompass one loop and eight loops, respectively, the ground truth is based on a template. For the World datasets [see Fig. 6(c) and (d)], the ground truth is generated by simulating a vehicle driving over a sphere. For these datasets, we have precise control over when a loop is detected, and for how many poses, a loop extends back in time. With this, we can detailedly investigate the differences in accuracy of COP-SLAM and of G^{2}O for increasing numbers of loops (5, 25, 100, 500) and for weak and strong overlap between loops, resulting in eight different datasets. For example, Fig. 6(c) shows typical results obtained when using five weakly overlapping loops and Fig. 6(d) when using 500 strongly overlapping loops.

For each of the 1000 pose-chains and for each of the edges in a pose-chain, including all loop-closing edges, a different anisotropic 6×6 covariance matrix is generated. In the 3×3 submatrices, related to translation and to rotation, respectively,

TABLE I
BINOCULAR VISUAL-SLAM BENCHMARK DATASETS (SORTED ON LENGTH OF TRAJECTORY)

Dataset	Length	Overlap	Frames	FPS	Resolution	HFOV	Baseline	Ground truth
Pittsburg C	27 km	12%	110 k	30 (auto key framing)	640x480	43°	12 cm	GPS
Pittsburg B	14 km	10%	57 k	30 (auto key framing)	640x480	43°	12 cm	GPS
Pittsburg A	8 km	31%	40 k	30 (auto key framing)	640x480	43°	12 cm	GPS
KITTI VO 02	5 km	15%	5 k	10	1241x376	80°	54 cm	RTK-GPS + IMU
KITTI VO 00	4 km	38%	5 k	10	1241x376	80°	54 cm	RTK-GPS + IMU
New College	3 km	96%	52 k	20	512x384	66°	12 cm	GPS (poor reception)

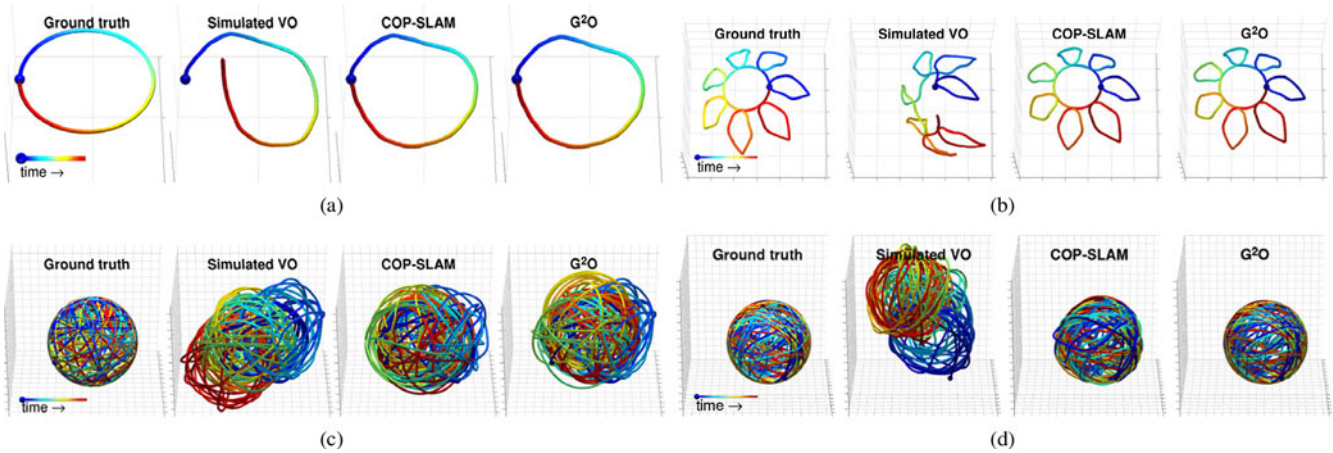

Fig. 6. Results of both methods obtained on one of the 100 simulated trajectories for simulated datasets. Loop is shown in (a), Flower in (b), World with five weakly overlapping loops in (c), and World with 500 strongly overlapping loops in (d). From left to right: ground truth, simulated visual odometry, stratified COP-SLAM, G^2O.

the ratio between the largest eigenvalue of the submatrix and its smallest eigenvalue is, on average, 100. This is similar to what we observe for covariance matrices obtained by linear error propagation inside real binocular front-ends. According to the covariance matrices, random error terms are sampled and added to the ground-truth edges (this is all performed using Lie group methods). Although the errors per edge are relatively small, the accumulated error (drift) they cause in nodes is significant. This can be observed in Fig. 6 and is similar to what we have observed for results of real binocular front-ends. For the simulated datasets, the ground truth of each node is known exactly. Therefore, no ground-truth interpolation or prealignment is required, as is done for our real-world binocular and monocular datasets.

Besides our own simulated datasets, we also use the well-known Sphere dataset [9]; see Fig. 8. With its 2500 nodes and its 9800 edges, it is a densely connected pose-graph and the exact opposite of a pose-chain. We use this dataset to investigate COP-SLAM's performance outside the domain for which it is designed. The combination of our extensive and realistic datasets, including 60 km of binocular data, together with the different performance metrics and the different vision-based front-ends that we use, set a challenging standard for pose-chain visual-SLAM evaluation.

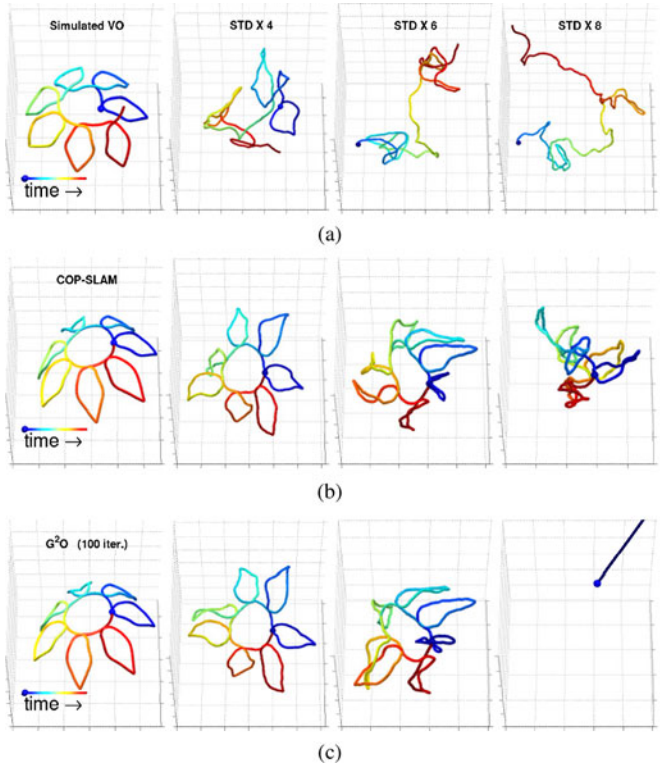

Fig. 7. Results of both methods on the Flower dataset for increasing VO noise. Simulated VO estimates are shown in (a), results of COP-SLAM in (b), and results of G^2O in (c).

D. Results Simulated Data

We start with a detailed analysis of the accuracy and efficiency of COP-SLAM and of G^2O on our simulated datasets, which are

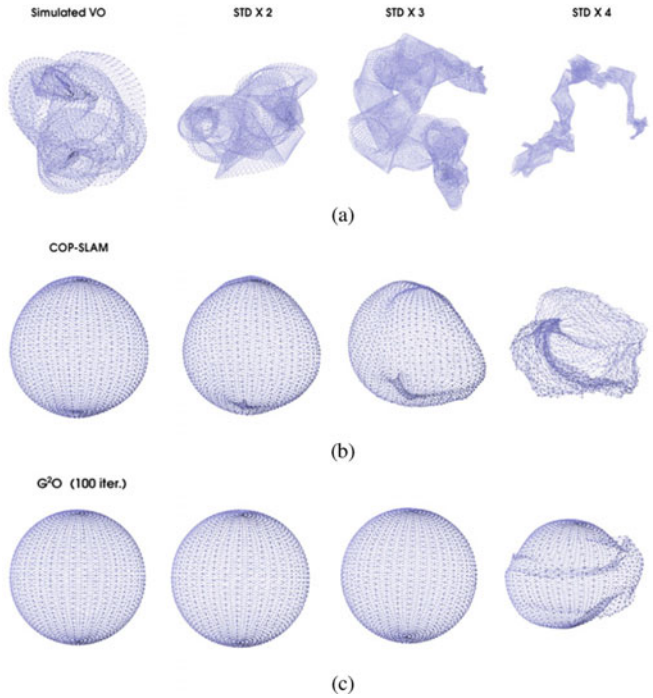

Fig. 8. Results of both methods on the publicly available Sphere benchmark dataset [9] for increasing VO noise. Simulated VO estimates are shown in (a), results of COP-SLAM in (b), and results of G^2O in (c).

conceptually similar to our stereo benchmark datasets. These are the Loop, Flower, World-5, and World-25 datasets, which have relatively few loops and little overlap between loops. COP-SLAM's optimality conditions are investigated next, for the different solution spaces R3, SO(3), SE(3), and SIM(3). After this, we analyze the influence of increasing the number of loops (i.e., 5, 25, 100, and 500 loops), as well as the influence of weakly and strongly overlapping loops. Finally, we demonstrate COP-SLAM's guaranteed convergence property by subjecting it to a degradation test. For this experiment, we use the densely connected Sphere pose-graph dataset [9].

For all experiments, the edges of a simulated pose-chain together with their covariance matrices are sequentially fed to COP-SLAM and to G^2O, thereby mimicking an online SLAM system. As COP-SLAM can only deal with isotropic uncertainty for the rotational and translational subspaces, we compute the average variance from the two 3×3 submatrices related to the rotational and translational subspaces. Please note that the computed variances are different for each edge in the pose-chain, depending on how accurate each edge is expected to be. G^2O uses the original covariance matrices and does take all statistical information into account, giving it an extra advantage over COP-SLAM for all our simulated experiments.

1) Accuracy and Efficiency: By setting the number of iterations used by the Gauss–Newton optimizer of G^2O, we can trade off accuracy and efficiency. In our first experiment, we, therefore, investigate the speed of convergence of monolithic COP-SLAM, stratified COP-SLAM and of G^2O using 1, 2, 3, 5, 10, and 100 Gauss–Newton iterations after each loop detection. The performance metrics averaged over all 100 trajectories of all four datasets (Loop, Flower, World-5, and World-25 datasets) are shown in Fig. 9. The reported timing values only include time spent on optimization and not time spent on file IO.

It can be observed that both COP-SLAM approaches (monolithic and stratified) converge in a single step to a level of accuracy that is similar to that of G^2O, while needing a factor 100 less computation time. The stratified COP-SLAM approach slightly outperforms the monolithic COP-SLAM approach both in accuracy and efficiency, as expected.

For the absolute position performance metric, it takes G^2O on average four iterations to become more accurate than COP-SLAM. For the absolute orientation metric, there are two iterations, and for the Chi square metric, there are three iterations. When using a single Gauss–Newton iteration, the accuracy of G^2O is significantly less than that of COP-SLAM for all performance metrics, while G^2O's computation time is significantly higher than that of COP-SLAM.

Table II provides a more detailed look into the absolute position performance metric (all other performance metrics show the same behavior). The reported values provide this performance metric averaged over all 100 experiments for each dataset individually. The standard deviation and the error relative to the error of the simulated visual odometer are also provided. Furthermore, we now also show the computation time of stratified COP-SLAM as a percentage of the computation time of G^2O. It can be observed that the accuracy of COP-SLAM versus G^2O is very dataset dependent. Also note that the difference in accuracy between COP-SLAM and G^2O relative to the initial error is marginal. Even when G^2O uses 100 iterations after each loop detection, the differences are only a few percentages of the initial error. In contrast, the computation time of G^2O is significantly higher than that of COP-SLAM, even when using only three iterations.

For all experiments for which their results are summarized in Table II, G^2O has converged to a point in the solution space that is close to the global minimum within four Gauss–Newton iterations. However, iterative optimizers like Gauss–Newton offer no guarantees that a satisfactory solution is obtained within a fixed number of iterations. To illustrate this, we have performed an additional experiment in which we kept simulating trajectories, until the result of G^2O when using four iterations was higher than the initial error of the simulated visual odometer. The first trajectory for which this has occurred is shown in Fig. 11, along with the results of G^2O and of G^2O initialized by COP-SLAM. It can be seen that only when using COP-SLAM as an initializer, G^2O provides satisfactory result within four Gauss–Newton iterations for this specific experiment.

An alternative would be to use an adaptive number of Gauss–Newton iterations inside G^2O and to guarantee that a solution is returned with a value for the cost function that is less than that of the initialization. This would require the use of *line search* or *trust region* methods inside G^2O. We observed that using such a trust region method (i.e., the Levenberg–Marquardt optimizer of G^2O) increases G^2O's computation time several orders of magnitude for our datasets, whereas the overhead of using COP-SLAM as an initializer is shown to be marginal.

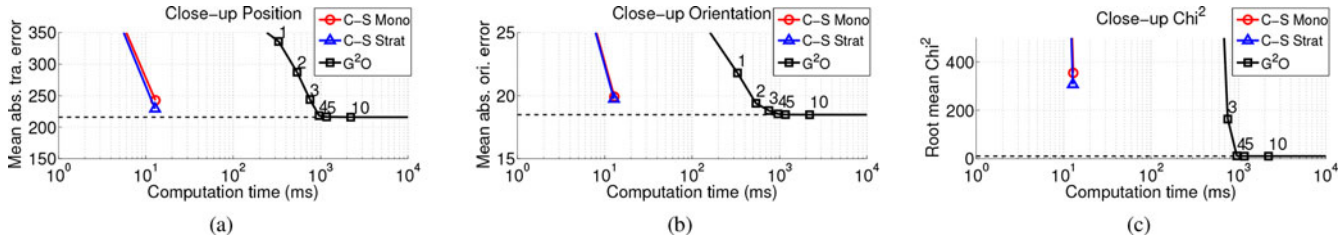

Fig. 9. Convergence, averaged over all synthetic datasets, of monolithic COP-SLAM (COP-SLAM Mono) in red, stratified COP-SLAM (COP-SLAM Strat) in blue, and G^{2}O in black. For G^{2}O, its convergence is shown when using 1, 2, 3, 4, 5, 10, and 100 iterations after each loop detection. The initial errors are 600 m, 47°, and 1.31×10^3, for the position, orientation, and Chi^2 metrics, respectively.

TABLE II
RESULTS ON SIMULATED DATA AVERAGED OVER 100 EXPERIMENTS

			Position Error (m)				Time (ms)		
	Poses	Loops	VO	COP-SLAM Mono	COP-SLAM Strat	G^{2}O (3 iter.)	COP-SLAM Mono	COP-SLAM Strat	G^{2}O (3 iter.)
Loop	10000	1	117 ± 40	63 ± 26 (54.1%)	**53 ± 21 (45.3%)**	75 ± 72 (63.9%)	8	**8 (3.8%)**	215
Flower	8120	8	252 ± 88	43 ± 11 (16.9%)	39 ± 11 (15.3%)	**35 ± 10 (14.0%)**	14	**14 (1.6%)**	855
World 5	4026	5	1008 ± 204	532 ± 154 (52.8%)	**499 ± 99 (49.5%)**	589 ± 236 (58.5%)	6	**6 (1.9%)**	316
World 25	4026	25	1025 ± 216	332 ± 65 (32.4%)	326 ± 64 (31.8%)	**276 ± 57 (26.9%)**	24	**23 (1.4%)**	1644
			Position Error (m)				Time (ms)		
	Poses	Loops	VO	COP-SLAM Mono	COP-SLAM Strat	G^{2}O (4 iter.)	COP-SLAM Mono	COP-SLAM Strat	G^{2}O (4 iter.)
Loop	10000	1	117 ± 40	63 ± 26 (54.1%)	53 ± 21 (45.3%)	**50 ± 20 (43.0%)**	8	**8 (3.1%)**	269
Flower	8120	8	252 ± 88	43 ± 11 (16.9%)	39 ± 11 (15.3%)	**35 ± 10 (14.0%)**	14	**14 (1.3%)**	1089
World 5	4026	5	1008 ± 204	532 ± 154 (52.8%)	**499 ± 99 (49.5%)**	512 ± 108 (50.8%)	6	**6 (1.5%)**	406
World 25	4026	25	1025 ± 216	332 ± 65 (32.4%)	326 ± 64 (31.8%)	**276 ± 57 (26.9%)**	24	**23 (1.1%)**	2082
			Position Error (m)				Time (ms)		
	Poses	Loops	VO	COP-SLAM Mono	COP-SLAM Strat	G^{2}O (100 iter.)	COP-SLAM Mono	COP-SLAM Strat	G^{2}O (100 iter.)
Loop	10000	1	117 ± 40	63 ± 26 (54.1%)	53 ± 21 (45.3%)	**49 ± 19 (42.0%)**	8	**8 (0.2%)**	5245
Flower	8120	8	252 ± 88	43 ± 11 (16.9%)	39 ± 11 (15.3%)	**35 ± 10 (14.0%)**	14	**14 (0.1%)**	24137
World 5	4026	5	1008 ± 204	532 ± 154 (52.8%)	**499 ± 99 (49.5%)**	503 ± 102 (49.9%)	6	**6 (0.1%)**	8642
World 25	4026	25	1025 ± 216	332 ± 65 (32.4%)	326 ± 64 (31.8%)	**276 ± 57 (26.9%)**	24	**23 (0.1%)**	45486

G^{2}O using 3, 4, and 100 iterations.

TABLE III
ERROR ON SIMULATED DATA AVERAGED OVER 100 EXPERIMENTS (ERRORS ARE RELATIVE TO INITIAL ERROR)

		R3		SO(3)		SE(3)		SIM(3)	
		isotropic	anisotropic	isotropic	anisotropic	isotropic	anisotropic	isotropic	anisotropic
Position error	COP-SLAM	**55.89%**	58.66%	na.	na.	42.10%	42.74%	17.52%	21.81%
	G^{2}O (100 iter.)	**55.89%**	**58.62%**	na.	na.	**35.03%**	**35.63%**	**17.55%**	**20.86%**
Orientation error	COP-SLAM	na.	na.	**58.37%**	59.46%	57.17%	58.91%	62.11%	62.39%
	G^{2}O (100 iter.)	na.	na.	**58.37%**	**59.44%**	**48.46%**	**50.84%**	**57.36%**	**59.03%**
Chi^2 error	COP-SLAM	**0.02%**	0.11%	**0.08%**	0.63%	2.47%	12.21%	2.73%	16.96%
	G^{2}O (100 iter.)	**0.02%**	**0.02%**	**0.08%**	**0.08%**	**0.004%**	**0.004%**	**0.004%**	**0.004%**

2) Optimality: The proof for COP-SLAM's optimality (see Theorem 2) states that when the solution space allows for a bi-invariant metric and when noise in relative pose estimates is isotropic, then COP-SLAM provides an optimal solution to the pose-graph problem for singe loops. In other words, under these restricted conditions, it provides an optimal solution to the pose-graph problem that is similar to that of G^{2}O. When we violate these conditions, i.e., add an-isotropic noise and no bi-invariant metric is available, then COP-SLAM cannot provide optimal solutions, whereas G^{2}O can. To get more insight in these aspects, we simulated a single loop, consisting of 1000 poses, in the solution spaces R3, SO(3), SE(3), and SIM(3), and we experimented with isotropic and anisotropic noise. The results averaged over 100 experiments are provided in Table III.

Both R3 and SO(3) allow for a bi-invariant metric. When we look at their respective columns for isotropic noise in Table III, then COP-SLAM and G^{2}O provide the same optimal results, as expected. When adding anisotropic noise, we can see minor difference between the accuracy of COP-SLAM and G^{2}O. When looking at solution spaces that do not have a bi-invariant

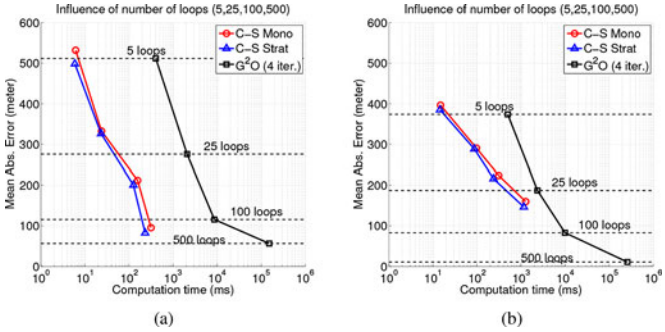

Fig. 10. Effect of the number of loop closures and the overlap between loops on the absolute position performance metric. The left figure (a) shows results for increasing numbers of weakly overlapping loops, and the right figure (b) shows the results for increasing numbers of strongly overlapping loops. The error of the visual odometer for these experiments is 1008 m.

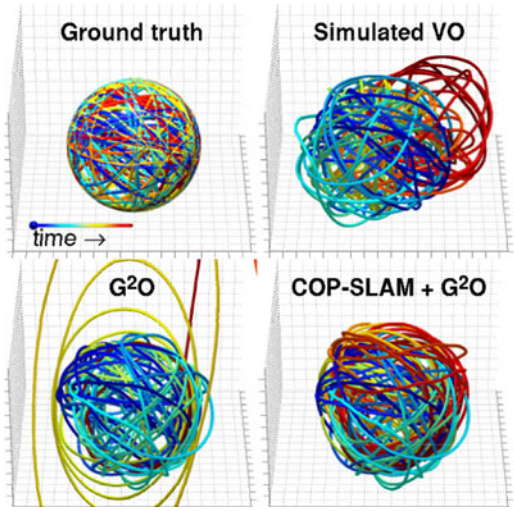

Fig. 11. Simulated trajectory for which G^2O did not converge within four iterations. From left to right: ground truth, result of G^2O using four iterations, and result of COP-SLAM followed by G^2O using four iterations.

metric, i.e., SE(3) and SIM(3), we see that the differences in accuracy between COP-SLAM and G^2O are increased. It is important to observe that although COP-SLAM can only provide optimal solutions to the pose-graph problem under restricted conditions, it always returns a solution that exhibits a significant reduction in error. Furthermore, for pose-chains, its reduction in error is also close to that of G^2O. This is due to its provable property of convergence and we come back to this in Section IV-D4. Before that, we first consider another type of violation of COP-SLAM's optimality conditions, i.e., multiple loops.

3) Graph Density and Topology: In this experiment, we evaluate the effect of increasing the number of loop closures (graph density) and increasing the overlap between loops (graph topology). For this, we use the four World datasets with 5, 25, 100, and 500 loops, respectively. We generate 100 test trajectories for each four datasets with weak overlap between loops and 100 test trajectories with strong overlap between loops. As the number of successive edges in the pose-chain is the same for all World datasets, increasing the number of weakly overlapping loops implies that the length of loops must reduce. For strongly overlapping loops, this is not the case. The results for the absolute position performance metric are shown in Fig. 10 (all other performance metrics show the same behavior).

The first thing that can be observed is that G^2O's accuracy improves when increasing the number of loops and also when increasing the overlap between loops. Clearly, this is as expected from a nonlinear iterative pose-graph optimizer. COP-SLAM behaves differently, however. While its accuracy also improves with the number of loops, its accuracy does not necessarily improve when increasing the overlap between loops. When there are few loops, e.g., 5, then it is better for COP-SLAM to have strong overlap. When there are many loops, e.g., 500, then it is better for COP-SLAM to have weak overlap between loops.

This behavior can be explained from its theoretical properties, which were provided in Section II-D2. COP-SLAM provides subspace optimal solutions for single loops. For COP-SLAM, it is, therefore, ideal that every node is part of one loop or only a few loops. Consequently, when there are only few loops, then they should be long, such that every node is at least part of one loop. However, when there are many loops, they should be shorter to prevent nodes being part of too many loops.

4) Graceful Degradation: The proof for COP-SLAM's convergence (see Theorem 1) states that COP-SLAM will return a solution in which loops are closed, regardless of the error in relative pose estimates. This implies that COP-SLAM exhibits graceful degradation.

To demonstrate this, we have performed two experiments where we increase the relative pose error far beyond what is typically encountered for real-world datasets. In one experiment, we use a typical weakly connected pose-chain (see Fig. 7), and in the other, we use the densely connected Sphere pose-graph (see Fig. 8). For the pose-chain in Fig. 7, we can clearly see that COP-SLAM always closes the loops, but its results gets more distorted, as the noise is increased. For G^2O, we observe similar performance, but for the highest noise level, it got stuck in a local minima. When applying the same test to the densely connected Sphere pose-graph, we see in Fig. 8 that G^2O can, as expected, take advantage of the increase in graph edges. Nevertheless, COP-SLAM can even for this densely connected pose-graph close all loops and reduce the error significantly.

This is a remarkable result, as COP-SLAM's computation time for the Sphere dataset is only 77 ms and does not depend on the noise level. G^2O's computation time for the highest noise level is 40 s, when using batch processing with 100 Gauss–Newton iterations and 7 min when using online processing with one Gauss–Newton iteration per detected loop.

To conclude our simulations, we would like to point out that although COP-SLAM is less accurate than G^2O for densely connected pose-graphs, it provably provides solutions in which loops are closed. The relative performance difference between COP-SLAM and G^2O simply increases with the number of loops and with more overlap between loops. This behavior is as expected, because more overlapping loops makes the pose-graph problem more nonlinear. Although there are robotic tasks in environments that will result in densely connected pose-graphs,

TABLE IV
RESULTS ON 60 KILOMETERS OF BINOCULAR TRAJECTORIES

			Position Error (m)			Orientation Error (degrees)			Time (ms)	
	Poses	Loops	VO	COP-SLAM Strat	G^2O (4 iter.)	VO	COP-SLAM Strat	G^2O (4 iter.)	COP-SLAM Strat	G^2O (4 iter.)
Pittsburgh A	6927	14	15	**6** (40.0%)	**6** (40.0%)	n/a	n/a	n/a	**49** (1.6%)	2972
Pittsburgh B	10 396	6	249	**49** (19.7%)	50 (20.0%)	n/a	n/a	n/a	**30** (1.4%)	2100
Pittsburgh C	19 268	20	422	**37** (8.8%)	**37** (8.8%)	n/a	n/a	n/a	**166** (1.8%)	9056
KITTI 00	4541	9	48	**13** (27.1%)	**13** (27.1%)	14	4 (28.6%)	**2** (14.3%)	**24** (1.8%)	1336
KITTI 02	4661	3	54	33 (61.1%)	**26** (48.1%)	10	6 (60.0%)	**5** (50.0%)	**9** (1.3%)	693
New College	52 480	98	n/a	n/a	n/a	n/a	n/a	n/a	**1531** (1.6%)	94928

there are also many tasks and environments that will results in weakly connected pose-graphs, i.e., result in pose-chains. When this is the case, COP-SLAM is an excellent light-weight alternative to iterative non-linear optimizers like G^2O. We emphasize this further with experiments using challenging real-world binocular data.

E. Results Binocular Data

In this section, we evaluate the real-world applicability of COP-SLAM, by quantitatively comparing its accuracy to that of G^2O. For this, we use our own datasets (Pittsburgh A,B,C) and the two longest KITTI benchmark datasets that have loops (see Table I). A qualitative comparison is also provided on the New College dataset. The Pittsburgh datasets are processed with our visual odometer [23], and the KITTI and New College datasets are processed with the publicly available LIBVISO2 [38]. For loop detection, we use RTAB-MAP [27]. The images of the benchmark datasets are sequentially fed to the SLAM front-end, whose estimated edges are again sequentially fed to COP-SLAM and to G^2O, thereby mimicking an online SLAM system.

Furthermore, the importance of the filter-like mechanism of COP-SLAM, discussed in Section II-B, is illustrated with a specific experiment. This is an important experiment as it shows that COP-SLAM can (to a certain extent) account for the statistical information of past loop closures, without the need to keep them as active edges in the pose-chain. This property is crucial when applying COP-SLAM to multiloop datasets.

1) Accuracy: The results of monolithic and stratified COP-SLAM and of G^2O on all six datasets are shown in Table IV. When a performance metric is not reported for a dataset, it is because no ground truth of sufficient quality is available for this dataset. For all datasets, the trajectories obtained by GPS, the visual-SLAM front-end, stratified COP-SLAM, and G^2O are visualized in Fig. 12 in 3-D.

Similar to our simulations, we observe that COP-SLAM provides satisfactory results that are similar to that of G^2O, while its computation time is significantly lower. Only from the values in Table IV, one can observe minor differences in accuracy between both methods, which are hardly observable in Fig. 12. Although no precise comparison is possible for the New College dataset, we can qualitatively observe that also for this dataset, which has significantly more loops, COP-SLAM provides satisfactory results that are similar to that of G^2O. In our view, COP-SLAM's results are of sufficient quality to be used in many contemporary and future robotic applications.

The results obtained on challenging binocular data, thereby underpinning those obtained on simulated data, show that COP-SLAM provides similar accuracy as G^2O on pose-chain datasets at a significant reduction in computation time. This observation allows for two interpretations: 1) COP-SLAM is able to use the most important local and global information contained in pose-chains; and 2) more general nonlinear iterative optimizers are an overkill when applied to pose-chains, as pose-chains are already relatively accurate at local scales. Clearly, these statements only apply to pose-chains and not to pose-graphs in general. Nevertheless, as pose-chains are a typical result of evermore widely used vision-based SLAM front-ends, we find them a relevant and promising SLAM representation.

2) Multiloop Filter-Like Behavior: In this experiment, we illustrate the importance of the filter-like behavior of COP-SLAM. This behavior is realized by the mechanism described in Section II-B, which updates the expected accuracy of edges, modeled by per-edge variances, after a loop is closed. The results on the Pittsburgh C dataset with and without using this filter-like mechanism, are shown in Fig. 13.

The critical utility of COP-SLAM's filter-like behavior is unmistakable. When it is not used, the results severely deteriorate. In this case, new loops can destroy previously closed loops. The updating of edge variances makes sure that the statistical information of previously closed loops is optimally accounted for when closing new loops. Conceptually, this mechanism makes an adaptive and optimal tradeoff between the statistical information of successive edges, of previous loops, and of the new loop.

F. Results Monocular Data

In this experiment, we demonstrate the extensibility of COP-SLAM as a back-end SLAM optimizer. To this purpose, we apply it to the novel solution space SIM(3) and extend it with a straightforward landmark updating mechanism (see Fig. 14).

Our SLAM front-end performs sliding-window bundle adjustment and provides estimates for the relative and absolute poses as well as for landmark locations. The difficulty is that the local scale of the reconstruction, provided by the SLAM front-end, will drift. This is a well-known nuisance of monocular SLAM and is typically caused by imperfections in the camera model and its calibrated parameters. Only by detecting loops, the scale drift becomes observable to the SLAM back-end and global optimization techniques, like [20], [25], are truly effective.

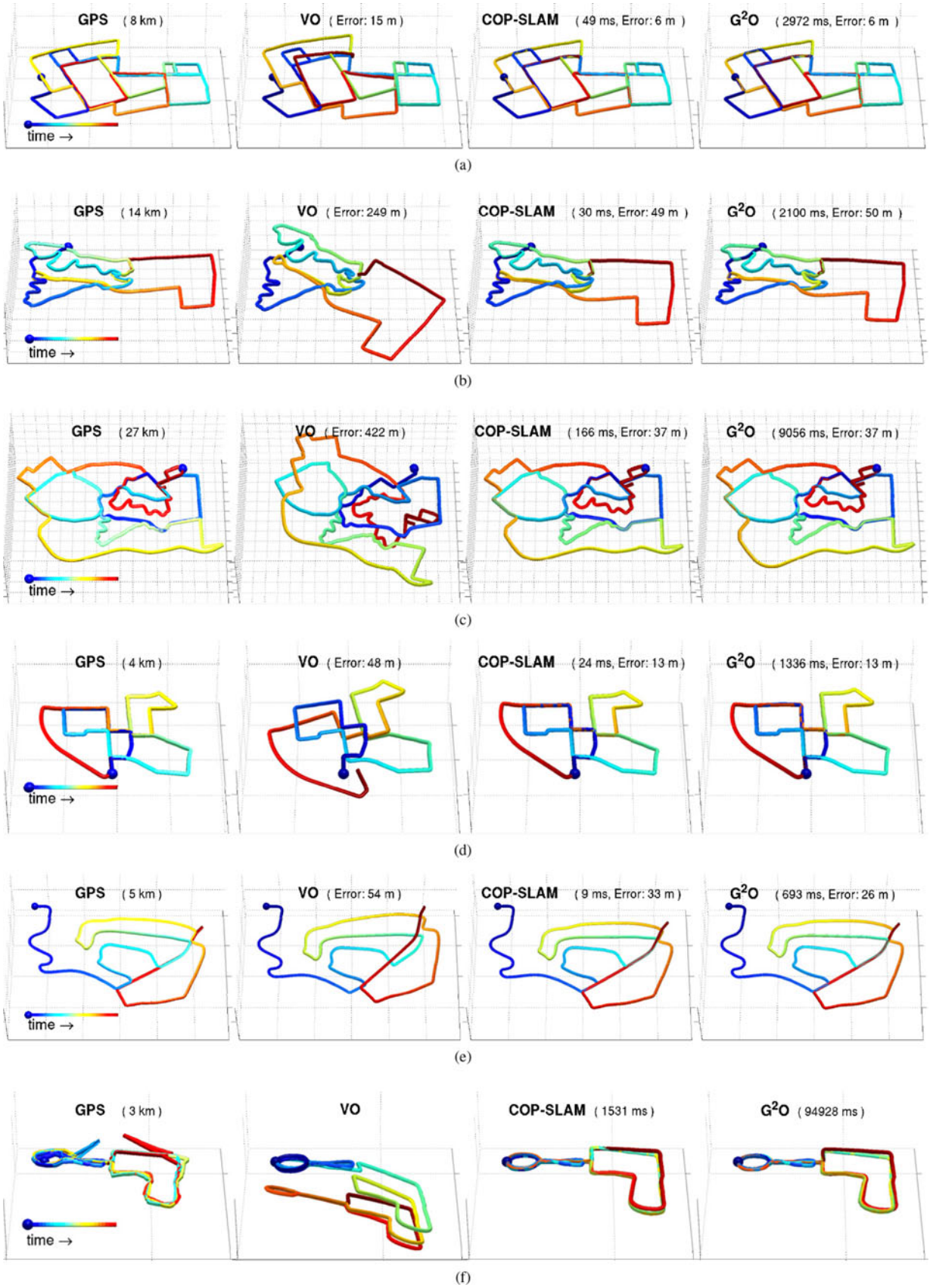

Fig. 12. Results plotted in 3-D for stratified COP-SLAM and for G^2O for all datasets. Pittsburgh A, Pittsburgh B, and Pittsburgh C are shown in (a)–(c), respectively, the KITTI O2 and KITTI 00 in (d) and (e), and the New College dataset in (f). The tile size in all figures is 200 m.

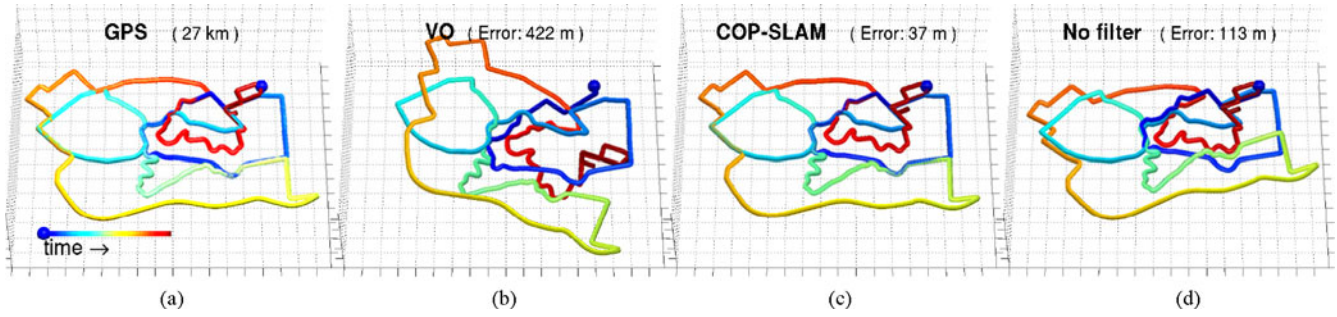

Fig. 13. Results of COP-SLAM on the Pittsburgh C dataset (c) with and (d) without using its per-edge variance (weight) updating mechanism, as described in Section II-B.

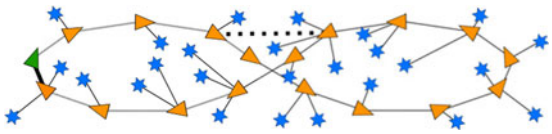

Fig. 14. Landmark modeling in COP-SLAM. The model is similar to that in Fig. 1(c) but with the addition of *landmark edges* between landmarks and absolute poses, which model the relative position of a landmark with respect to an absolute pose. Note that each landmark is only connected to one absolute pose, in our case that absolute pose at which the landmark was last observed. With this straightforward modeling, an improvement in absolute poses directly translates into an improvement of landmark positions.

Here, we apply COP-SLAM as the back-end optimizer, as an example of how it can be extended to other SLAM tasks. In order for COP-SLAM to close loops in the SIM(3) solution space, relative scale information must be explicitly available in loop-closing edges. The required relative scale transformation between the start and the end of a loop is estimated from efficient local map matching. One map is based on observations of loop-closing landmarks made at the start of the loop. The other map is based on observations made at the end of the loop. These maps are directly obtained from the sliding-window monocular odometer. The relative scale transformation then becomes observable by estimating the scale difference between these two local maps. COP-SLAM puts an error metric on this observable absolute scale difference, which is then minimized by altering the relative scale changes of successive edges.

In Fig. 15(a), we show the trajectory and map estimated by the monocular front-end. To illustrate that global optimization with scale drift is not a trivial task, Fig. 15(b) shows the result when using full bundle adjustment as a back-end optimizer. It is clear that it was not able to provide a satisfactory result, by reducing the value of its reprojection-based objective function from 1275×10^6 to 12×10^6, using 300 Levenberg–Marquardt iterations and taking 127 s. The satisfactory result obtained by stratified COP-SLAM in 1 ms is shown in Fig. 15(c). The value for the bundle adjustment objective function of COP-SLAM's solution is 232×10^3 and thereby significantly less than that of the SLAM front-end. When applying 100 iterations of bundle adjustment after COP-SLAM, the objective function reduced further to 9×10^3, and the result is shown in Fig. 15(d).

This experiment demonstrates the utility of COP-SLAM as a back-end SLAM optimizer. Its results will be usable *as is* for many robotic tasks or will be an excellent initialization for nonlinear optimization.

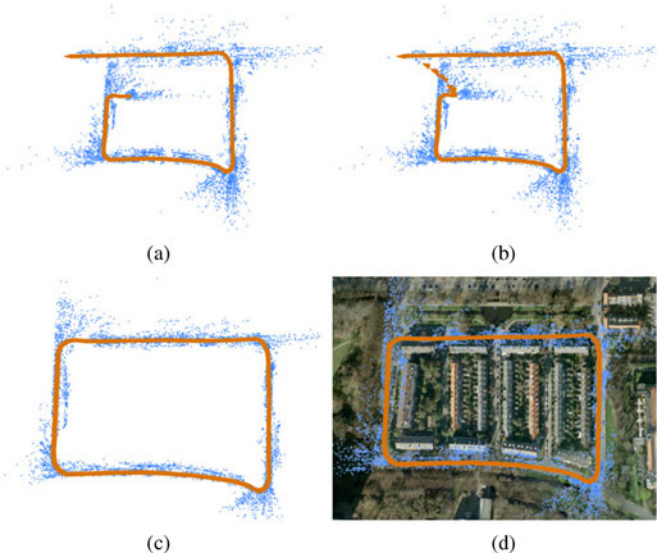

Fig. 15. Results on our 800-m-long monocular dataset. (a) Output of SLAM front-end. (b) Result after 300 iterations of full bundle adjustment. (c) Result of stratified COP-SLAM and (d) result when applying 100 iterations of full bundle adjustment after stratified COP-SLAM.

V. Conclusion

We have presented a detailed analysis of the recently proposed COP-SLAM method. It is a theoretically unique SLAM approach that is specifically designed to optimize pose-chains, which are extremely sparse locally accurate pose-graphs that result from applying state-of-the-art visual-SLAM front-ends in large-scale environments for which frequent loop detection is not desired or not possible.

At its core, COP-SLAM uses Lie group computational methods, whose convergence and optimality properties have been theoretically proven. It is highly extensible and can be applied to any Lie group for which its exponential and logarithmic maps can be computed, of which SE(2), SE(3), and SIM(3) are most relevant to robotics. On the basis of extensive real-world experiments, we can conclude that its theoretical properties, which allow it to compute its solution in closed form and in linear complexity in the number of edges, have minimal impact on its empirical performance. When applied to pose-chains, the difference between the accuracy of COP-SLAM and that of iterative optimizers, which are a factor 50–100 times more computationally demanding, is shown to be only a few percentages of the initial error. COP-SLAM's accuracy on pose-chain datasets

is thereby very close to that of maximum likelihood solutions provided by iterative optimizers.

We have shown that COP-SLAM has utility also outside the pose-chain domain. Its results for densely connected pose-graphs and for bundle adjustment show that it is a versatile method. In future work, it can easily be extended to other Lie groups for which exponential and logarithmic maps can be computed. This makes COP-SLAM an excellent lightweight SLAM back-end, when only limited computation resources are available. If adequate computation resources are available, it can be used as an initializer and thereby reduce the overall computation time and simultaneously increase the robustness and effectiveness of the robotic system.

REFERENCES

[1] J. J. Leonard and H. F. Durrant-Whyte, "Simultaneous map building and localization for an autonomous mobile robot," in *Proc. IEEE/RSJ Intell. Robots Syst.*, 1991, pp. 1442–1447.

[2] G. Dubbelman and B. Browning, "Closed-form online pose-chain SLAM," in *Proc. IEEE Int. Conf. Robot. Autom.*, Karlsruhe, Germany, 2013, pp. 5190–5197.

[3] H. Strasdat, J. M. M. Montiel, and A. J. Davison, "Real-Time monocular SLAM: Why filter?" in *Proc. IEEE Int. Conf. Robot. Autom.*, 2010, pp. 2657–2664.

[4] R. M. Eustice, H. Singh, and J. J. Leonard, "Exactly sparse delayed-state filters for view-based SLAM," *IEEE Trans. Robot.*, vol. 22, no. 6, pp. 1100–1114, Dec. 2006.

[5] C. Cadena and J. Neira, "SLAM in O(log(n)) with the combined kalman-information filter," *Robot. Auton. Syst.*, vol. 58, no. 11, pp. 1207–1219, Nov. 2010.

[6] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 1052–1067, Jun. 2007.

[7] P. Elinas, R. Sim, and J. J. Little, "σSLAM: Stereo vision SLAM using the Rao-Blackwellised particle filter and a novel mixture proposal distribution," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2006, pp. 1564–1570.

[8] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA, USA: MIT Press, 2005.

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

[10] G. Grisetti, C. Stachniss, and W. Burgard, "Non-linear constraint network optimization for efficient map learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 3, pp. 428–439, Sep. 2009.

[11] N. Sünderhauf and P. Protzel, "Towards a robust back-end for pose graph SLAM," in *Proc. Int. Conf. Robot. Autom.*, 2012, pp. 1254–1261.

[12] R. Wagner, O. Birbach, and U. Frese, "Rapid development of manifold-based graph optimization systems for multi-sensor calibration and SLAM," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2011, pp. 3305–3312.

[13] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *Int. J. Robot. Res.*, vol. 32, pp. 216–235, Feb. 2012.

[14] E. Olson, J. Leonard, and S. Teller, "Fast iterative alignment of pose graphs with poor initial estimates," in *Proc. Int. Conf. Robot. Autom.*, May, 2006, pp. 2262–2269.

[15] M. K. Grimes, D. Anguelov, and Y. Yann, "Hybrid hessians for flexible optimization of pose graphs," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2010, pp. 2997–3004.

[16] Y. Latif, C. Cadena, and J. Neira, "Robust loop closing over time," presented at the Robot. Sci. Syst. Conf., Sydney, Australia, 2012.

[17] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid, "RSLAM: A system for large-scale mapping in constant-time using stereo," *Int. J. Comput. Vision*, vol. 94, no. 2, pp. 198–214, Jun. 2010.

[18] Y.-D. Jian, D. C. Balcan, and F. Dellaert, "Generalized subgraph preconditioners for large-scale bundle adjustment," in *Proc. Int. Conf. Comput. Vision*, 2011, pp. 295–302.

[19] K. Konolige and M. Agrawal, "FrameSLAM: From bundle adjustment to real-time visual mapping," *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 1066–1077, Oct. 2008.

[20] H. Strasdat, A. J. Davison, J. M. M. Montiel, and K. Konolige, "Double window optimisation for constant time visual SLAM," in *Proc. Int. Conf. Comput. Vision*, 2011, pp. 2352–2359.

[21] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recog.*, 2004, vol. 1, pp. 652–659.

[22] Z. Zhu, T. Oskiper, S. Samarasekera, R. Kumar, and H. S. Sawhney, "Ten-fold improvement in visual odometry using landmark matching," in *Proc. IEEE Int. Conf. Comput. Vision*, Oct. 2007, pp. 1–8.

[23] G. Dubbelman, W. van der Mark, and F. C. A. Groen, "Accurate and robust ego-motion estimation using expectation maximization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nice, France, 2008, pp. 3914–3920.

[24] J. Civera, D. R. Bueno, A. J. Davison, and J. M. M. Montiel, "Camera self-calibration for sequential Bayesian structure from motion," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2009, pp. 403–408.

[25] H. Strasdat, J. M. M. Montiel, and A. J. Davison, "Scale drift-aware large scale monocular SLAM," presented at the Robot. Sci. Syst., Zaragoza, Spain, 2010.

[26] M. Cummins and P. Newman, "Appearance-only SLAM at large scale with FAB-MAP 2.0," *Int. J. Robot. Res.*, vol. 30, pp. 1100–1123, Nov., 2010.

[27] M. Labbé and L. Michaud, "Memory management for real-time appearance-based loop closure detection," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2011, pp. 1271–1276.

[28] M. Bosse and R. Zlot, "Place recognition using regional point descriptors for 3D mapping," in *Field and Service Robotics* Springer Tracts in Advanced Robotics), vol. 62. New York, NY, USA: Springer, 2010, pp. 195–204.

[29] G. Dubbelman, I. Esteban, and K. Schutte, "Efficient trajectory bending with applications to loop closure," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Taipei, Taiwan, 2010, pp. 1–7.

[30] G. Dubbelman, P. Hansen, B. Browning, and M. B. Dias, "Orientation only loop-closing with closed-form trajectory bending," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2012, pp. 815–821.

[31] P. Newman, D. Cole, and K. Ho, "Outdoor SLAM using visual appearance and laser ranging," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2006, pp. 1180–1187.

[32] C. Stachniss, U. Frese, and G. Grisetti. (2007). OpenSLAM. [Online]. Available: http://www.openslam.org

[33] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981.

[34] R. Raguram, O. Chum, M. Pollefeys, J. Matas, and J. Frahm, "USAC: A universal framework for random sample consensus," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 2022–38, Aug. 2013.

[35] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.

[36] L. Carlone, P. Aragues, J. A. Castellanos, and B. Bona, "A fast and accurate approximation for planar pose graph optimization," *Int. J. Robot. Res.*, vol. 33, no. 6, pp. 1–23, May 2014.

[37] L. Zhao, S. Huang, and G. Dissanayake, "Linear SLAM: A linear solution to the feature-based and pose graph SLAM based on submap joining," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Tokyo, Japan, 2013, pp. 24–30.

[38] A. Geiger, J. Ziegler, and C. Stiller, "StereoScan: Dense 3D reconstruction in real-time," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2011, pp. 963–968.

[39] X. Pennec, "Intrinsic statistics on Riemannian manifolds: Basic tools for geometric measurements," *J. Math. Imag. Vision*, vol. 25, no. 1, pp. 127–154, Jul. 2006.

[40] B. C. Hall, *Lie Groups, Lie Algebras, and Representations: An Elementary Introduction*. New York, NY, USA: Springer, 2003.

[41] M. P. do Carmo, *Riemannian Geometry* (ser. Mathematics: Theory and Applications), 1st ed., Wallach, Ed. Boston, MA USA: Birkhäuser, Jan. 1992.

[42] G. Dubbelman and B. Browning. (2015). "COP-SLAM: Proofs of optimality and convergence," Eindhoven Univ. Technol., Eindhoven, The Netherlands, Tech. Rep. [Online]. Available: http://www.gijsdubbelman.com/copslam-tech-report

[43] F. C. Park, "Distance metrics on the rigid-body motions with applications to mechanism design," *Trans. ASME*, vol. 117, pp. 48–54, Mar. 1995.

[44] G. Dubbelman, "Intrinsic statistical techniques for robust pose estimation," Ph.D. dissertation, Univ. Amsterdam, Amsterdam, The Netherlands, 2011.
[45] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner, "On measuring the accuracy of SLAM algorithms," *Auton. Robots*, vol. 27, no. 4, pp. 387–407, Sep. 2009.
[46] G. Dubbelman, P. Hansen, and B. Browning, "Bias compensation in visual odometry," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, no. 3, pp. 2828–2835.
[47] M. Smith, I. Baldwin, W. Churchill, R. Paul, and P. Newman, "The new college vision and laser data set," *Int. J. Robot. Res.*, vol. 28, no. 5, pp. 595–599, May 2009.

Gijs Dubbelman (S'07–M'11) received the B.Sc. degree in information and communication technology, the M.Sc. degree (*cum laude*) in artificial intelligence, and the Ph.D. degree in 2011 on the topic of intrinsic statistical techniques for robust pose estimation, all from University of Amsterdam, Amsterdam, The Netherlands.

He is a Researcher with Eindhoven University of Technology, Eindhoven, The Netherlands. In 2011 and 2012, he was a Member of the Field Robotics Center, Carnegie Mellon's Robotics Institute, where he researched 3-D computer vision for autonomous robots.

Brett Browning (M'02) received the B.Electrical Engineer and B.Sc. (Math) degrees in 1996, and the Ph.D. degree in 2000, all from University of Queensland, St. Lucia, Australia.

He is currently a Senior Systems Scientist with the School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, where he has been a Faculty Member of the Robotics Institute since 2002. His research interests include robot autonomy and, in particular, real-time robot perception, applied machine learning, and teamwork.