Hierarchical Optimization on Manifolds for Online 2D and 3D Mapping

Giorgio Grisetti Rainer Kümmerle Cyrill Stachniss Udo Frese Christoph Hertzberg

Abstract— **In this paper, we present a new hierarchical optimization solution to the graph-based simultaneous localization and mapping (SLAM) problem. During online mapping, the approach corrects only the coarse structure of the scene and not the overall map. In this way, only updates for the parts of the map that need to be considered for making data associations are carried out. The hierarchical approach provides accurate non-linear map estimates while being highly efficient. Our error minimization approach exploits the manifold structure of the underlying space. In this way, it avoids singularities in the state space parameterization. The overall approach is accurate, efficient, designed for online operation, overcomes singularities, provides a hierarchical representation, and outperforms a series of state-of-the-art methods.**

I. Introduction

A popular way to address the simultaneous localization and mapping (SLAM) problem are "graph-based" approaches [4, 6, 8, 9, 10, 12, 14, 18, 16]. In this formulation, the poses of the robot are modeled by nodes in a graph. Spatial constraints between poses that result from observations and from odometry are encoded in the edges between the nodes. In graph-based SLAM, two problems need to be addressed: First, constraints need to be extracted from sensor data. This is referred to as the SLAM front-end. Second, given the constraints, the most likely configuration as well as the pose uncertainty need to be computed. This is referred to as the SLAM back-end or as the optimization engine. During online operation, these two problems are typically solved in an alternating way.

Addressing the first problem, requires solving the data association problem, i.e. determining if the current measurement covers the same part of the environment as previous observations. To avoid a comparison to all previously perceived observations, efficient SLAM systems bound the search for correspondences. One way to achieve this is to limit the search area based on the current uncertainty estimate of the robot. One should note that it is not required to correct the whole graph for making data associations. It is typically sufficient to have a corrected estimate of the robot's surroundings and the area in which the robot might be given its pose uncertainty estimate. To identify this area, an accurate estimate of the coarse structure of the environment is sufficient – the whole map is not needed.

The contribution of this paper is two-fold: First, we present a novel and highly effective optimization approach based on Gauss-Newton with sparse Cholesky factorization that considers a manifold representation of the state space. The concept of a manifold allows us to estimate 3D rotations with Gauss-Newton without running into parameterization singularities, as they occur for instance with Euler angles. This leads to a more robust and highly accurate error minimization approach. Second, we present a novel SLAM back-end that aims at efficiently and accurately estimating the coarse structure of the environment for online mapping. This is done by using a hierarchical approach which can be seen as "lazy" in each step because instead of repeatedly optimizing all nodes of the graph, it computes a solution to a simplified problem. This simplified problem, however, contains all relevant information needed by the front-end to operate successfully and it is constructed incrementally. Our hierarchy consists of multiple, sparse pose-graphs representing the environment, see Figure 1 for an illustration.

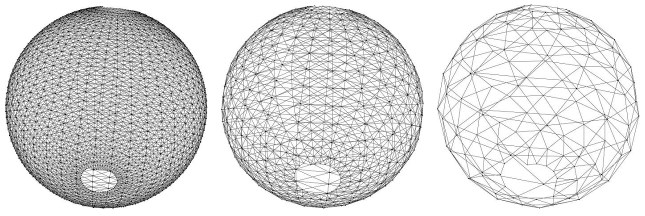
Fig. 1. This figure shows a 3 level hierarchy constructed by our approach while optimizing a 3D network of 2,000 nodes and 8,647 constraints. Left: the lower level representing the original problem. Middle: the intermediate level and right, the top level.

The different levels of the hierarchy represent the original problem at different levels of abstraction. The bottom level corresponds to the original input, while the higher levels capture the structural information of the environment in a compact manner. Every time a new observation is obtained, only the highest level needs to be optimized completely. When a higher level of the hierarchy is modified, only the regions of the map which are substantially changed are updated in the lower levels. In this way, we limit the computational requirements while preserving the global consistency.

It is worth mentioning that our sparsification procedure is an accurate non-linear approximation and accordingly, one can compute the covariances of the nodes by considering the sparse graph only. This enables the front-end to operate efficiently and to use popular approaches for data association like the χ^2 test or the joint compatibility test [15]. We validate our approach on simulated and real world 2D as well as 3D datasets and provide comparisons to state-of-the-art approaches including TreeMap [5], TORO [8, 9], and Gauss-Newton with sparse Cholesky factorization.

This work has partly been supported by the DFG under SFB/TR-8 as well as by the European Commission under FP7-231888-EUROPA and FP7-248258-First-MM.

G. Grisetti, C. Stachniss, and R. Kümmerle are with the University of Freiburg. U. Frese and C. Hertzberg are with the University of Bremen.

II. Related Work

There is a large variety of SLAM approaches available in the robotics community and we mainly focus on graph-based approaches here. Lu and Milios [14] were the first to refine a map by globally optimizing the system of equations to reduce the error introduced by constraints. Gutmann and Konolige [10] proposed an effective way for constructing such a network and for detecting loop closures while running an incremental estimation algorithm. Since then, many approaches for minimizing the error in the constraint network have been proposed. For example, Howard *et al.* [12] apply relaxation to localize the robot and build a map. Frese *et al.* [6] propose a variant of Gauss-Seidel relaxation called multi-level relaxation (MLR). It applies relaxation at different resolutions. Olson *et al.* [18] presented an efficient optimization approach which is based on the stochastic gradient descent and can efficiently correct even large pose-graphs. Grisetti *et al.* proposed an extension of Olson's approach that uses a tree parameterization of the nodes in 2D and 3D. In this way, they speed up the convergence [9].

Also the ATLAS framework [1] is related to our approach. It constructs a two-level hierarchy and employs a Kalman filter to construct the bottom level. Then, a global optimization approach aligns the local maps at the second level. Similar to ATLAS, Estrada *et al.* proposed Hierarchical SLAM [3] as a technique for using independent local maps. In case of place re-visiting, these maps are joined or augmented. In contrast to their work, we use a non-linear sparsification approach which leads to sparse representations at different levels of abstraction, all being consistent. In addition to that, we propose a new optimization approach that avoids singularities in the non-Euclidean space of rotations.

Most optimization techniques focus on computing the best map given the constraints and are called SLAM back-ends. In contrast to that, SLAM front-ends seek to interpret the sensor data to obtain the constraints that are the basis for the optimization approaches. Olson [17], for example, presented a front-end with outlier rejection based on spectral clustering. For making data associations in the SLAM front-ends statistical tests such as the χ^2 test or joint compatibility test [15] are often applied. The work of Nüchter *et al.* [16] aims at building an integrated SLAM system for 3D mapping. The main focus lies on the SLAM front-end for finding constraints. For optimization, a variant of the approach of Lu and Milios [14] for 3D settings is applied. All these front-ends also apply a global optimization procedure to compute a consistent map. Combining them with the approach presented in this paper will make them more efficient.

III. Map Learning using Pose-Graphs

Throughout this paper, we consider the SLAM problem in its graph-based formulation. The poses of the robot are described by the nodes of a graph and edges between these nodes represent spatial constraints between them. The edges are constructed from observations or from odometry.

A complete graph-based SLAM system has to address different problems, namely, constructing the abstract graph representation from the raw sensor measurements (front-end) and computing the most likely configuration of the poses (back-end), often including uncertainty estimates. To correctly construct the graph, the front-end typically requires a consistent estimate of the structure of the environment together with the expected uncertainty for data associations.

A. The SLAM Front-end

Our front-end is able to construct 2D and 3D maps, from laser data. Every time the robot travels a minimum distance a new node is added to the graph. Edges connecting the current node and the previous one are added by scan-matching. To detect loop closures, our approach computes a quick approximation of the conditional covariances of all nodes in the graph, conditioned on the current robot position. Then, a scan-matching procedure is applied for every node whose 3σ ellipse intersects the current pose and a loop-closing edge is added if the matching succeeds. To reject false closures, we use a spectral-clustering approach which determines the maximally consistent set of constraints around the current robot location. All in all, our approach is an own implementation of the front-end described by Olson [17] but extended to 3D.

B. The SLAM Back-end

The goal of such graph-based mapping algorithms is to find the configuration of the nodes that maximizes the likelihood of the observations. Let $\mathbf{x} = (\mathbf{x}_1, \ \dots \ , \mathbf{x}_n)^T$ be a vector of parameters, where $\mathbf{x}_i$ describes the pose of node i. Let $\mathbf{z}_{ij}$ and $\boldsymbol{\Omega}_{ij}$ be respectively the mean and the information matrix of an observation of node j seen from node i, perturbed by Gaussian noise. Let $\mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij})$ be a function that computes a difference between the expected observation of the node $\mathbf{x}_j$ seen from the node $\mathbf{x}_i$ and the real observation $\mathbf{z}_{ij}$ gathered by the robot. For simplicity of notation, we will encode the indices of the measurement in the indices of the error function

$$\mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij}) \overset{\text{def.}}{=} \mathbf{e}_{ij}(\mathbf{x}_i, \mathbf{x}_j) \overset{\text{def.}}{=} \mathbf{e}_{ij}(\mathbf{x}). \tag{1}$$

Let $\mathcal{C}$ be the set of pairs of indices for which a constraint (observation) $\mathbf{z}$ exists. The goal of a maximum likelihood approach is to find the configuration of the nodes $\mathbf{x}^*$ that minimizes the negative log likelihood $\mathbf{F}(\mathbf{x})$ of all the observations

$$\mathbf{F}(\mathbf{x}) = \sum_{\langle i,j \rangle \in \mathcal{C}} \underbrace{\mathbf{e}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{e}_{ij}}_{\mathbf{F}_{ij}} \tag{2}$$

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \, \mathbf{F}(\mathbf{x}). \tag{3}$$

Thus, it seeks to solve Eq. (3).

IV. Pose-Graph Optimization on a Manifold

This section describes the first contribution of this paper, an accurate and efficient way of optimizing a pose-graph, namely solving Eq. (3) and estimating the involved uncertainty estimates. In brief, our approach applies Gauss-Newton on a manifold using sparse Cholesky factorization.

Considering that the state space is not a Euclidean vector space, a manifold allows us to appropriately handle the singularities introduced by the angular components. The concept of manifolds enables us to find a better linearization of the system and thus leads to an efficient and accurate solution. In the remainder of this section, we first describe how to compute the solution to Eq. (3) via iterative linearizations. Then, we modify this solution by considering the concept of manifolds.

A. Error Minimization via Iterative Local Linearizations

If a good initial guess $\breve{\mathbf{x}}$ of the robot's poses is known, the numerical solution of Eq. (3) can be obtained by using the popular Gauss-Newton or Levenberg-Marquardt algorithms. The idea is to approximate the error function by its first order Taylor expansion around the current initial guess $\breve{\mathbf{x}}$

$$\begin{aligned} \mathbf{e}_{ij}(\breve{\mathbf{x}}_i + \mathbf{\Delta x}_i, \breve{\mathbf{x}}_j + \mathbf{\Delta x}_j) &= \mathbf{e}_{ij}(\breve{\mathbf{x}} + \mathbf{\Delta x}) && (4) \\ &\simeq \mathbf{e}_{ij} + \mathbf{J}_{ij}\mathbf{\Delta x}. && (5) \end{aligned}$$

Here $\mathbf{J}_{ij}$ is the Jacobian of $\mathbf{e}_{ij}(\mathbf{x})$ computed in $\breve{\mathbf{x}}$ and $\mathbf{e}_{ij} \overset{\text{def.}}{=} \mathbf{e}_{ij}(\breve{\mathbf{x}})$. Substituting Eq. (5) in the error terms $\mathbf{F}_{ij}$ of Eq. (2), we obtain:

$$\begin{aligned} & \mathbf{F}_{ij}(\breve{\mathbf{x}} + \mathbf{\Delta x}) && (6) \\ &= \mathbf{e}_{ij}(\breve{\mathbf{x}} + \mathbf{\Delta x})^T \mathbf{\Omega}_{ij} \mathbf{e}_{ij}(\breve{\mathbf{x}} + \mathbf{\Delta x}) && (7) \\ &\simeq (\mathbf{e}_{ij} + \mathbf{J}_{ij}\mathbf{\Delta x})^T \mathbf{\Omega}_{ij} (\mathbf{e}_{ij} + \mathbf{J}_{ij}\mathbf{\Delta x}) && (8) \\ &= \underbrace{\mathbf{e}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{e}_{ij}}_{\mathrm{c}_{ij}} + 2\underbrace{\mathbf{e}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{J}_{ij}}_{\mathbf{b}_{ij}} \mathbf{\Delta x} + \mathbf{\Delta x}^T \underbrace{\mathbf{J}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{J}_{ij}}_{\mathbf{H}_{ij}} \mathbf{\Delta x} && (9) \\ &= \mathrm{c}_{ij} + 2\mathbf{b}_{ij}\mathbf{\Delta x} + \mathbf{\Delta x}^T \mathbf{H}_{ij} \mathbf{\Delta x} && (10) \end{aligned}$$

With this local approximation, we can rewrite the function $\mathbf{F}(\mathbf{x})$ given in Eq. (2) as

$$\begin{aligned} \mathbf{F}(\breve{\mathbf{x}} + \mathbf{\Delta x}) &= \sum_{\langle i,j\rangle \in \mathcal{C}} \mathbf{F}_{ij}(\breve{\mathbf{x}} + \mathbf{\Delta x}) && (11) \\ &\simeq \sum_{\langle i,j\rangle \in \mathcal{C}} \mathrm{c}_{ij} + 2\mathbf{b}_{ij}\mathbf{\Delta x} + \mathbf{\Delta x}^T \mathbf{H}_{ij} \mathbf{\Delta x} && (12) \\ &= \mathrm{c} + 2\mathbf{b}^T \mathbf{\Delta x} + \mathbf{\Delta x}^T \mathbf{H} \mathbf{\Delta x}. && (13) \end{aligned}$$

The quadratic form in Eq. (13) is obtained from Eq. (12) by setting $\mathrm{c} = \sum \mathrm{c}_{ij}$, $\mathbf{b} = \sum \mathbf{b}_{ij}$, and $\mathbf{H} = \sum \mathbf{H}_{ij}$. It can be minimized in $\mathbf{\Delta x}$ by solving the linear system

$$\mathbf{H}\,\mathbf{\Delta x}^* = -\mathbf{b}. \qquad (14)$$

The matrix $\mathbf{H}$ is the information matrix of the system and is sparse by construction, having non-zeros between poses connected by a constraint. Its number of non-zero blocks is twice the number of constrains plus the number of nodes. This allows to solve Eq. (14) by sparse Cholesky factorization. An highly efficient implementation of sparse Cholesky factorization can be found in the library CSparse [2].

The linearized solution is then obtained by adding to the initial guess the computed increments

$$\mathbf{x}^* = \breve{\mathbf{x}} + \mathbf{\Delta x}^*. \qquad (15)$$

The popular Gauss-Newton algorithm iterates the linearization in Eq. (13), the solution in Eq. (14), and the update step in Eq. (15). In every iteration, the previous solution is used as the linearization point and the initial guess.

The procedure described above is a general approach to multivariate function minimization, here derived for the special case of the SLAM problem. The general approach, however, assumes that the space of parameters $\mathbf{x}$ is Euclidean, which is not valid for SLAM. This may lead to sub-optimal solutions.

B. Linearization on a Manifold

To cope with the fact that in SLAM the state space is not Euclidean, we propose to apply the error minimization on a manifold. A manifold is a mathematical space that is not necessarily Euclidean on a global scale, but can be seen as Euclidean on a local scale [13].

In the context of our SLAM problem, each parameter $\mathbf{x}_i$ consists of a translation vector $\mathbf{t}_i$ and a rotational component α_i. The translation $\mathbf{t}_i$ clearly forms a Euclidean space. In contrast to that, the rotational components α_i span over the non-Euclidean 2D or 3D rotation group $SO(2)$ or $SO(3)$. To avoid singularities, these spaces are usually described in an overparameterized way, e.g., by rotation matrices or quaternions. Directly applying Eq. (15) to these overparameterized representations de-normalizes the angles and thus invalidates the configuration which then introduces errors in the solution. To overcome this problem, one can use a minimal representation for the angles (like Euler angles in 3D). This, however, is then subject to singularities.

An alternative idea is to consider the underlying space as a manifold and to define an operator $\boxplus$ that maps a local variation $\mathbf{\Delta x}$ in the Euclidean space to a variation on the manifold, $\mathbf{\Delta x} \mapsto \mathbf{x} \boxplus \mathbf{\Delta x}$. We refer the reader to [11, §1.3] for more mathematical details. With this operator, a new error function can be defined as

$$\begin{aligned} \mathbf{e}_{ij}^{\breve{\mathbf{x}}}(\mathbf{\Delta}\tilde{\mathbf{x}}_i, \mathbf{\Delta}\tilde{\mathbf{x}}_j) &\overset{\text{def.}}{=} \mathbf{e}_{ij}(\breve{\mathbf{x}}_i \boxplus \mathbf{\Delta}\tilde{\mathbf{x}}_i, \breve{\mathbf{x}}_j \boxplus \mathbf{\Delta}\tilde{\mathbf{x}}_j) && (16) \\ &= \mathbf{e}_{ij}(\breve{\mathbf{x}} \boxplus \mathbf{\Delta}\tilde{\mathbf{x}}) \simeq \mathbf{e}_{ij} + \tilde{\mathbf{J}}_{ij}\mathbf{\Delta}\tilde{\mathbf{x}} && (17) \end{aligned}$$

where $\breve{\mathbf{x}}$ spans over the original over-parameterized space. In our approach, we use quaternions. The term $\mathbf{\Delta}\tilde{\mathbf{x}}$ is a small increment around the original position $\breve{\mathbf{x}}$ expressed in a minimal representation. Here, we use rotation axis scaled by the rotation angle.

In more detail, we represent the increments $\mathbf{\Delta}\tilde{\mathbf{x}}$ as 6D vectors $\mathbf{\Delta}\tilde{\mathbf{x}}^T = (\mathbf{\Delta}\tilde{\mathbf{t}}^T\ \mathbf{\Delta}\tilde{\alpha}^T)$, where $\mathbf{\Delta}\tilde{\mathbf{t}}$ denotes the translation and $\Delta\alpha^T = (\Delta\alpha_x\,\Delta\alpha_y\,\Delta\alpha_z)^T$ is the axis-angle representation of the 3D rotation. Conversely, $\breve{\mathbf{x}}^T = (\breve{\mathbf{t}}^T\ \breve{\mathbf{q}}^T)$ uses a quaternion $\breve{\mathbf{q}}$ to encode the rotational part. Thus, the operator $\boxplus$ can be expressed by first converting $\mathbf{\Delta}\alpha$ to a quaternion $\mathbf{\Delta q}$ and then applying the transformation $\mathbf{\Delta x}^T = (\mathbf{\Delta t}^T\ \mathbf{\Delta q}^T)$ to $\breve{\mathbf{x}}$. In the equations describing the error minimization, these operations can nicely be encapsulated by the $\boxplus$ operator. The Jacobian $\tilde{\mathbf{J}}_{ij}$ can be expressed by

$$\tilde{\mathbf{J}}_{ij} = \left.\frac{\partial \mathbf{e}_{ij}(\breve{\mathbf{x}} \boxplus \mathbf{\Delta}\tilde{\mathbf{x}})}{\partial \mathbf{\Delta}\tilde{\mathbf{x}}}\right|_{\mathbf{\Delta}\tilde{\mathbf{x}}=\mathbf{0}}. \qquad (18)$$

With a straightforward extension of the notation, we can insert Eq. (17) in Eq. (8) and Eq. (11). This leads to the following increments:

$$\tilde{\mathbf{H}}\,\boldsymbol{\Delta}\tilde{\mathbf{x}}^* \quad = \quad -\tilde{\mathbf{b}}. \tag{19}$$

Since the increments $\boldsymbol{\Delta}\tilde{\mathbf{x}}^*$ are computed in the local Euclidean surroundings of the initial guess $\breve{\mathbf{x}}$, they need to be re-mapped into the original redundant space by the $\boxplus$ operator. Accordingly, the update rule of Eq. (15) becomes

$$\mathbf{x}^* \quad = \quad \breve{\mathbf{x}} \boxplus \boldsymbol{\Delta}\tilde{\mathbf{x}}^*. \tag{20}$$

In summary, formalizing the minimization problem on a manifold consists of first computing a set of increments in a local Euclidean approximation around the initial guess by Eq. (19) and second accumulating the increments in the global non-Euclidean space by Eq. (20).

V. Hierarchical Pose-Graph

The second contribution of this paper is a hierarchical pose-graph. It allows us to accurately model the coarse structure of the environment online. This information is *essential* for making good data associations in the SLAM front-end.

The key idea of the hierarchical pose-graph is to represent the problem at different levels of abstraction. Each level is a pose-graph and there are connections modeling correspondences between abstraction levels. The lowest level ($k = 0$) represents the original input. Each node at level $k > 0$ represents a sub-graph at level $k - 1$. An edge between two nodes at level $k > 0$ models the constrains between the sub-graphs and can be computed analytically as will be explained below. It is obvious that the higher the level of abstraction, the lower the number of parameters to describe the environment, and thus the lower the quality of the representation at that level but the faster the optimization.

More formally, we represent the problem using a hierarchy of K graphs. Let $\mathcal{G}^{[k]}$ be the graph at level k. The graph $\mathcal{G}^{[k]}$ consists of a set of nodes $\{\mathbf{x}_i^{[k]}\}$ and a set of edges $\{e_{ij}^{[k]}\}$. Each node $\mathbf{x}_i^{[k]}$ at level k is associated to

(i) a "representative" node $\mathbf{x}_i^{[k-1]}$ at level $k-1$ and
(ii) a connected sub-graph $\mathcal{G}_i^{[k-1]}$ at level $k-1$.

An edge $e_{ij}^{[k]}$ between the nodes $\mathbf{x}_i^{[k]}$ and $\mathbf{x}_j^{[k]}$ at level $k > 0$ exists if the two sub-graphs $\mathcal{G}_i^{[k-1]}$ and $\mathcal{G}_j^{[k-1]}$ are connected. Figure 2 illustrates a simple two layered graph structure. The idea is to construct a high level graph by partitioning the lower level in local maps, represented by the sub-graphs $\{\mathcal{G}_i^{[k-1]}\}$. Each local map is then represented by a node at the higher level. Edges between nodes at the high level encode the relations between local maps arising from the connectivity between neighboring local maps.

A. Construction of the Hierarchy

To build the graph $\mathcal{G}^{[k]}$ at level k from the graph $\mathcal{G}^{[k-1]}$ at level $k-1$, it is sufficient to define groups of connected nodes in $\mathcal{G}^{[k-1]}$. In our current implementation, we group the nodes based on a straight-forward threshold criterion that

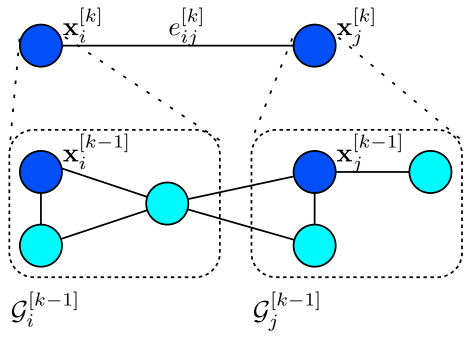

Fig. 2. Simple two layered graph structure. Every node $\mathbf{x}_i^{[k]}$ in the higher level corresponds to a connected sub-graph $\mathcal{G}_i^{[k-1]}$ at lower level and to a node $\mathbf{x}_i^{[k-1]}$ within the sub-graph. An edge $e_{ij}^{[k]}$ exists if two sub-graphs at low level are connected.

considers the distance on the graph. Note that this worked well in all our settings, but may offer room for further improvements. Let these groups be $\{\mathcal{G}_i^{[k-1]}\}$. For each group $\mathcal{G}_i^{[k-1]}$, we choose a representative node $\mathbf{x}_i^{[k-1]}$ for $\mathcal{G}_i^{[k-1]}$. This representative becomes the node $\mathbf{x}_i^{[k]}$ at level k.

To obtain a consistent hierarchy, we have to add an edge $e_{ij}^{[k]}$ between $\mathbf{x}_i^{[k]}$ and $\mathbf{x}_j^{[k]}$ at level k if the corresponding sub-graphs are connected. This edge has to capture the information encoded in all edges of $\mathcal{G}_i^{[k-1]}$ and $\mathcal{G}_j^{[k-1]}$ as well as all edges connecting both.

Accordingly, we need to compute a mean $\mathbf{z}_{ij}^{[k]}$ and information matrix $\boldsymbol{\Omega}_{ij}^{[k]}$ for the edge $e_{ij}^{[k]}$. The parameters depend only on the configuration of the sub-graphs $\mathcal{G}_i^{[k-1]}$ and $\mathcal{G}_j^{[k-1]}$ and are computed as follows.

Let $\mathcal{G}_{i\cup j}^{[k-1]}$ be the union of the graphs $\mathcal{G}_i^{[k-1]}$ and $\mathcal{G}_j^{[k-1]}$ and their interconnecting edges. We can obtain the relative position of $\mathbf{x}_j^{[k-1]}$ with respect to $\mathbf{x}_i^{[k-1]}$ and thus the mean $\mathbf{z}_{ij}^{[k]}$ of $e_{ij}^{[k]}$ by optimizing $\mathcal{G}_{i\cup j}^{[k-1]}$, while forcing $\mathbf{x}_i^{[k-1]}$ to the origin.

Let $\mathbf{H}_{i\cup j}^{[k-1]}$ be the information matrix of $\mathcal{G}_{i\cup j}^{[k-1]}$ computed according to Section IV during the optimization of this sub-graph. Since $\mathbf{x}_i^{[k-1]}$ lies in the origin, the covariance matrix $\boldsymbol{\Sigma}_j^{[k]}$ of the node $\mathbf{x}_j^{[k-1]}$ is equal to the one of the edge $e_{ij}^{[k]}$ and can be obtained by extracting the corresponding block of $(\mathbf{H}_{i\cup j}^{[k-1]})^{-1}$. Since $\mathbf{H}_{i\cup j}^{[k-1]}$ is sparse by construction, this procedure can be carried out efficiently. Given the covariance of the edge, the information matrix is obtained directly by $\boldsymbol{\Omega}_{ij}^{[k]} = (\boldsymbol{\Sigma}_j^{[k]})^{-1}$.

B. Extending the Hierarchical Pose-Graph

As the robot moves through the environment, information has to be added to the hierarchical pose-graph. This is done by adding a node and corresponding edges to the bottom level of the hierarchy as done in standard graph-based SLAM. According to a distance-based threshold criterion, the newly created node is either added to an existing group or it becomes the representative of a new one at level 0. This procedure it recursively executed upwards the hierarchy until no new groups need to be created. Edges are added and its parameters are updated accordingly.

C. Hierarchical Graph Optimization

After an update of the hierarchical pose-graph, an optimization is carried out. The optimization always starts at the top level using the manifold optimization approach presented in Section IV. As a result, all nodes at the highest level are updated. However, changes are only propagated to lower levels if the optimization leads to significant changes in the node configurations. These changes are detected by monitoring the difference between each node $\mathbf{x}_i^{[k]}$ and its representative $\mathbf{x}_i^{[k-1]}$ at level $k-1$. Whenever the distance between $\mathbf{x}_i^{[k]}$ and $\mathbf{x}_i^{[k-1]}$ exceeds a given threshold (in our current implementation: 0.05 m or 2 deg), we propagate the changes downwards. This is achieved by applying a rigid body transformation to each subgraph $\mathcal{G}_i^{[k-1]}$ so that $\mathbf{x}_i^{[k]} = \mathbf{x}_i^{[k-1]}$. When required by the front-end, we generate a locally consistent estimate of a portion of the map by optimizing the corresponding sub-graph at low level. During the optimization, we impose the additional constraints $\mathbf{x}_i^{[k-1]} = \mathbf{x}_i^{[k]}$. In this way, we account for the estimate at the higher level in the lower level. After the mapping process one may consider to run a last optimization at level $k = 0$ to obtain the best possible map.

It should be noted that if one focuses only on *offline* mapping with *given* data association and starting from a *good initial guess*, the hierarchy is not needed since optimizing the original input will provide the desired solution.

VI. Experiments

The experiments are designed to show

1) that the manifold optimization approach is able to find a better configuration of the nodes compared to all other techniques evaluated here.
2) that the hierarchical pose-graph is able to efficiently compute consistent estimates (mean and covariance) at all levels.
3) that our approach outperforms the other state-of-the-art techniques in terms of run-time and can operate online.

To support our claims, we compared our approach to Gauss-Newton with sparse Cholesky factorization without manifolds (Section IV-A), TreeMap [5], and TORO with its incremental [8] and batch [9] version. We performed our test on 2D datasets (Intel research lab dataset and a simulated one) and 3D datasets (Stanford parking garage and a simulated sphere), see Figure 3. Both simulated datasets are the ones used in [9]. During all experiments, we use a three level hierarchy ($k = 0, 1, 2$). We provide an open-source implementation of our approach called "HOG-Man" written in C++ which is available online [7].

A. Manifold Optimization

In this first experiment, we evaluated how the optimization approach that considers the manifold improves the performance. Here, the complete set of constraints is provided to the optimizer and we measured the χ^2 error vs. runtime.

For this offline batch comparison, we used the simulated 3D dataset with a significant initial error and compared

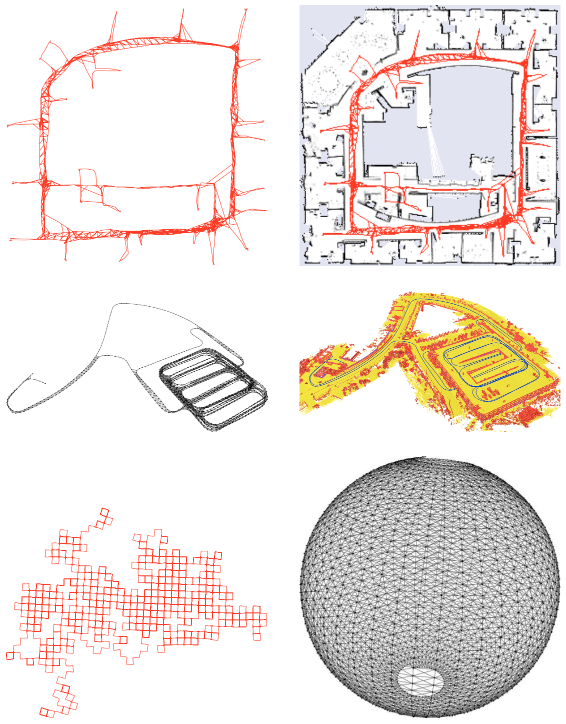

Fig. 3. The four datasets used in our experimental evaluation. Top row: pose-graph and grid map of the Intel research lab, middle row: pose-graph and 3D map of the Stanford parking garage, bottom left: simulated 2D dataset (W-10000), bottom right: simulated 3D dataset (sphere).

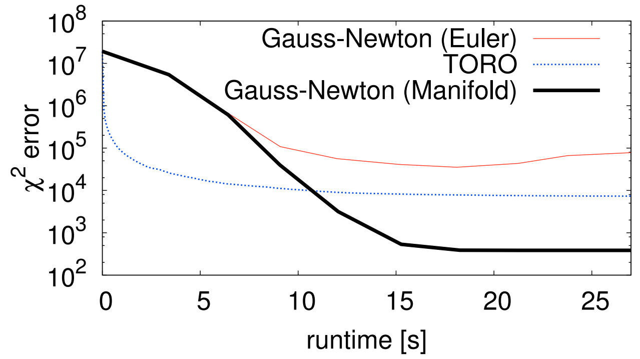

Fig. 4. Evolution of the χ^2 error for TORO, Gauss-Newton using Euler angles, and Gauss-Newton with manifold on the 3D Sphere dataset.

the results of Gauss-Newton with and without the manifold linearization, i.e. here by using Euler angles. We furthermore provide a comparison to TORO [9]. Figure 4 depicts the results. As can be seen, not considering the singularities appropriately can lead to significant errors. Also TORO quickly converges to a visually good-looking solution but still leaves space for improvements. Even after performing a large number of iterations, the remaining χ^2 error of TORO is significantly larger than the one of our new approach. This caused by the fact that TORO is an approximative approach that assumes roughly spherical covariances and optimizes translations and rotations separately.

B. Consistency of the Hierarchical Approach

The second experiment is designed to show that the sparsified pose-graphs (levels greater than 0) represents a good

TABLE I

Comparison of the 3σ covariance ellipses between the original problem and the levels of our hierarchy.

	Prob. mass not covered	Prob. mass outside
Intel	0.10%	10.18%
W-10000	2.53%	24.05%
Stanford	0.01%	7.88%
Sphere	2.75%	10.21%

TABLE II

Runtime comparison for the different approaches.

avg./std./max.[ms]	Our Approach	TreeMap [5]	TORO [9]
Intel	4 / 5 / 31	6 / 5 / 58	3 / 2 / 33
W-10000	25 / 20 / 183	1426 / 1342 / 9987	146 / 97 / 323
Stanford	25 / 26 / 189	2D pose graphs only	35 / 85 / 602
Sphere	89 / 42 / 213	2D pose graphs only	226 / 606 / 4615

approximation of the original problem. This is especially important for the SLAM front-end for efficiently making good data associations. Therefore, all experiments in the remainder of this section, are carried out online. This means that every time a new node is added to the graph, the optimization is carried out (incrementally).

The baseline of the comparison is thus the original problem, fully optimized without the hierarchical approach. To evaluate the quality of the most sparsified pose-graph (top level), we compare the Gaussian associated to each node of these graphs with the corresponding distribution of the original problem. We compute the probability mass within the 3σ bounds of the original problem that lies outside the same bound of the sparsified graph and vice versa.

Table I illustrates the results based on all data sets. As can be seen, the hierarchy approximates the original problem well. Especially, the probability mass that is not covered by the sparse pose-graphs (over-confident estimates) is around or below 0.1% for all real world datasets. In general, the uncertainty ellipses of the sparse graphs are typically bigger than the ones in the original problem (around 10% for the real world datasets).

C. Runtime Comparison

In the final experiment, we analyzed the runtime required by the different approaches to optimize the pose-graph. The results depicted in Table II show the average, the standard deviation, and the maximum runtime time of the optimization engine which was always executed after adding a new node. The timings are provided for all datasets analyzed in this paper. The experiment has been executed in a Core2Duo processor with a 2.4 GHz processor (single-thread).

As can be seen, our approach clearly outperforms TreeMap. Detailed investigation of TreeMap's tree data structure showed that heavy leaves in the tree, i.e. leaves with many poses, led to the poor performance. This is caused by revisited places leading to a fully connected clique of poses. Even worse, TreeMap combines several successive poses into one leaf during the first visit and has to add a duplicate pose to everyone of these after each revisit.

Our method furthermore outperforms TORO. In the comparably densely connected simulated pose-graphs, this effect was more prominent compared to the real world datasets.

VII. Conclusion

In this paper, we present a novel SLAM back-end for the graph-based SLAM systems. Its contribution is two-fold: First, an efficient optimization approach that takes the singularities of the angular components into account by considering a manifold when optimizing the pose-graph. This leads to a highly efficient and effective error minimization approach. Second, a hierarchical pose-graph that is able to model the problem at different levels of abstraction which can be optimized fast while providing support for making data associations. The overall approach is accurate, efficient, designed for online operation, overcomes singularities, provides a hierarchical representation, and outperforms a series of state-of-the-art methods.

References

[1] M. Bosse, P. M. Newman, J. J. Leonard, and S. Teller. An ATLAS framework for scalable mapping. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 1899–1906, 2003.

[2] T. A. Davis. *Direct Methods for Sparse Linear Systems*. SIAM Series on the Fundamentals of Algorithms. SIAM, Philadelphia, 2006.

[3] C. Estrada, J. Neira, and J.D. Tardós. Hierachical SLAM: Real-time accurate mapping of large environments. *IEEE Transactions on Robotics*, 21(4):588–596, 2005.

[4] J. Folkesson and H. Christensen. Graphical SLAM - a self-correcting map. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Orlando, FL, USA, 2004.

[5] U. Frese. Treemap: An $O(\log n)$ algorithm for indoor simultaneous localization and mapping. *Autonomous Robots*, 21(2):103–122, 2006.

[6] U. Frese, P. Larsson, and T. Duckett. A multilevel relaxation algorithm for simultaneous localisation and mapping. *IEEE Transactions on Robotics*, 21(2):1–12, 2005.

[7] G. Grisetti, R. Kümmerle, and C. Stachniss. The source code. http://www.openslam.org, 2009.

[8] G. Grisetti, D. Lodi Rizzini, C. Stachniss, E. Olson, and W. Burgard. Online constraint network optimization for efficient maximum likelihood map learning. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Pasadena, CA, USA, 2008.

[9] G. Grisetti, C. Stachniss, and W. Burgard. Non-linear constraint network optimization for efficient map learning. *IEEE Transactions on Intelligent Transportation Systems*, 2009. In press.

[10] J.-S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *Proc. of the IEEE Int. Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, 1999.

[11] C. Hertzberg. A framework for sparse, non-linear least squares problems on manifolds. Master's thesis, Univ. of Bremen, 2008.

[12] A. Howard, M.J. Matarić, and G. Sukhatme. Relaxation on a mesh: a formalism for generalized localization. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2001.

[13] J.M. Lee. *Introduction to Smooth Manifolds*, volume 218 of *Graduate Texts in Mathematics*. Springer Verlag, 2003.

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

[15] J. Neira and J.D. Tardós. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation*, 17(6):890–897, 2001.

[16] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. 6d SLAM with approximate data association. In *Proc. of the Int. Conference on Advanced Robotics (ICAR)*, pages 242–249, 2005.

[17] E. Olson. *Robust and Efficient Robotic Mapping*. PhD thesis, MIT, Cambridge, MA, USA, June 2008.

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