MAP ESTIMATION WITH LASER SCANS BASED ON INCREMENTAL TREE NETWORK OPTIMIZER

Dario Lodi Rizzini[1], *Stefano Caselli*[1]

[1]Università degli Studi di Parma
Dipartimento di Ingegneria dell'Informazione
viale G.P. Usberti 181A, 43100 Parma, Italy

ABSTRACT

Several adaptations of maximum likelihood approaches to incremental map learning have been proposed recently. In particular, an incremental network optimizer based on stochastic gradient descent provides a fast and easy-to-implement solution to the problem. In this paper, we illustrate a map builder that processes laser scans in order to extract the constraint network for the optimization algorithm. The map is stored in the form of a collection of scans corresponding to a subset of the poses of a robot moving in the environment. Such a map representation has the advantage of simplicity, but it requires careful processing of data associations. In particular, loop closures are performed in two steps: first, pose constraint candidates are searched and tested with a heuristic criterion; then, pairs of poses are computed through standard scan matching techniques. Our aim is to illustrate the effectiveness of a tree network optimizer integrated with a simple technique for data associations. Experiments reported in the paper show that a compact system integrating these two components works reasonably well with commonly used benchmarks.

1. INTRODUCTION

Several robotic applications require a sufficiently detailed representation of the environment to achieve specific tasks. Autonomously building maps of the environment has become a major problem in the robotics community. For this task the robot has at its disposal its motion information and its sensor observations, which are both affected by uncertainty. In literature, map building is often referred to as *Simultaneous Localization And Mapping (SLAM)*. To address SLAM different estimation techniques can be used, such as Kalman filters [1], Rao-Blackwellized particle filters [2] or maximum-likelihood (ML) estimation.

A recent approach to SLAM formulates mapping as a network of constraints. Relations of robot poses and observations are represented by constraints among nodes in a graphical model. Solution of the network corresponds to the estimation of the configuration of nodes minimizing least-square error. Lu and Milios [3] pioneered maximum likelihood approach proposing a brute force technique to align range scans. A numerically-safe variant of the algorithm with effective loop detection was presented in [4]. Other solutions are based on Gauss-Seidel relaxation [5, 6].

However, all these algorithms aiming to solve graph-based SLAM formulations are not fully suitable for online map estimation. Solving for the map requires all the data from the beginning: after each addition of constraints, the new estimation is built without re-using the map previously computed. Recently, incremental maximum-likelihood approaches have been presented. The incremental smoothing and mapping algorithm (*iSAM*) [7], which applies QR factorization to the information matrix to perform efficient back-substitution, directly updates matrices Q and R when new measurements are available. The treemap algorithm [8] performs efficient updates with a tree-based subdivision of the map into weakly-correlated components.

A stochastic gradient descent (SGD) technique has been proposed [9] to compute the configuration minimizing least-square error by using a representation which enables efficient updates. An incremental variant of the algorithm relies on several improvements such as tree parameterization, adaptive learning rate and selective update rules [10]. The resulting algorithm efficiently adjusts the poses given proper constraints and has the advantage of being straightforward to implement.

In this paper, we present a compact and self-contained implementation of a map solver based on the previously discussed online tree network optimizer. Several ML algorithms are not so easy to implement or depend on external libraries to perform complex matrix operations efficiently. The proposed map solver requires only an external scan matcher and can be implemented within few thousands lines of code. The system is naturally modular since it integrates the incremental optimizer with a constraint network builder. It takes odometric information and laser scans as an input and stores the map in the form of a collection of scans. Maps adopting a single scan as local patch have the advantage of being easy to implement, but they require a more careful procedure in detecting large cycles. Topo-

This work has been partially supported by "LARER – Laboratorio di Automazione della Regione Emilia-Romagna".

logical incorrectness due to false positives may cause bad misalignments during the optimization step. More reliable techniques exploiting multiple scans as local patches have been proposed for loop closure [4, 11]. However, our aim in this paper is to illustrate the effectiveness of a tree network optimizer integrated with a simple technique for data associations.

The paper is organized as follows. Following an outline of stochastic gradient descent–based map solvers, an incremental version of the tree network optimizer is illustrated in section 2. The construction of a graph from laser scans with matching and data association is discussed in section 3. Experiments are reported in section 4. Finally, conclusion remarks are given in section 5.

2. INCREMENTAL TREE NETWORK OPTIMIZER

The tree network optimizer (TORO) [12, 10] belongs to ML approaches, since its aim is to find the configuration of the nodes that maximizes the likelihood. Let $\mathbf{x} = (x_1 \ \cdots \ x_n)^T$ be a vector of parameters which describes a configuration of the nodes. Let δ_{ji} and Ω_{ji} be respectively the mean and the information matrix of an observation of node j seen from node i. Let $f_{ji}(\mathbf{x})$ be a function that computes a zero noise observation according to the current configuration of the nodes j and i.

Given a constraint between node j and node i, we define the *residual* r_{ji} introduced by the constraint as

$$r_{ji}(\mathbf{x}) \quad = \quad f_{ji}(\mathbf{x}) - \delta_{ji} \tag{1}$$

Let $\mathcal{C} = \{\langle j_1, i_1\rangle, \ldots, \langle j_M, i_M\rangle\}$ be the set of pairs of indices for which a constraint $\delta_{j_m i_m}$ exists. The aim is to find the configuration $\mathbf{x}^*$ minimizing the global error:

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x}} \sum_{\langle j,i\rangle \in \mathcal{C}} r_{ji}(\mathbf{x})^T \Omega_{ji} r_{ji}(\mathbf{x}). \tag{2}$$

2.1. Stochastic gradient descent with tree parameterization

In the tree network optimizer [12] the method used to compute Eq. (2) is *stochastic gradient descent*, that consists of minimizing the error by using the gradient evaluated on configuration variables $\mathbf{x}_i$ involved in the update of each constraint distinctly.

Error on each constraint $\langle j, i\rangle$ is minimized with an iterative gradient descent procedure. Gradient equation depends on the Jacobian J_{ji} of f_{ji}, on the information matrix Ω_{ji} and on residual r_{ji}:

$$\nabla \mathbf{x}_{ij} \quad \propto \quad \mathbf{H}^{-1} J_{ji}^T \Omega_{ji} r_{ji} \tag{3}$$

where $\mathbf{H}^{-1}$ is a preconditioning matrix. Details on other terms of gradient can be found in [9, 10]. The *incremental parameterization* introduced requires a reordering of nodes in the graph defined by a spanning tree. The initial node has null pose $\mathbf{p}_0$ and null parameter vector $\mathbf{x}_0$. The parameter $\mathbf{x}_i$ of every other node is the difference between the pose of the node and the pose of the parent node in the spanning tree:

$$x_i \quad = \quad p_i - p_{\mathrm{parent}(i)}, \tag{4}$$

where $\mathrm{parent}(i)$ refers to the index of parent of node i. In the original formulation [9] $\mathrm{parent}(i) = i - 1$, which corresponds to degeneration of the tree into a list.

To obtain the difference between two arbitrary nodes based on the tree, one needs to traverse the tree from the first node upwards to the first common ancestor of both nodes and then downwards to the second node. The same holds for computing the error of a constraint. We refer to the nodes one needs to traverse on the tree as the path of a constraint. For example, $\mathcal{P}_{ji}$ is the path from node i to node j for the constraint $\langle j, i\rangle$. The path can be divided into an ascending part $\mathcal{P}_{ji}^{[-]}$ of the path starting from node i and a descending part $\mathcal{P}_{ji}^{[+]}$ to node j. We can then compute the residual in the global frame by

$$r'_{ji} \quad = \quad \sum_{k^{[-]} \in \mathcal{P}_{ji}^{[-]}} x_{k^{[-]}} - \sum_{k^{[+]} \in \mathcal{P}_{ji}^{[+]}} x_{k^{[+]}} + R_i \delta_{ji}. \tag{5}$$

Here R_i is the homogeneous rotation matrix of the pose p_i.

Let $\Omega'_{ji} = R_i \Omega_{ji} R_i^T$ be the information matrix of a constraint in the global frame. According to [9], we compute an approximation of the Jacobian as

$$J'_{ji} \quad = \quad \sum_{k^{[+]} \in \mathcal{P}_{ji}^{[+]}} \mathcal{I}_{k^{[+]}} - \sum_{k^{[-]} \in \mathcal{P}_{ji}^{[-]}} \mathcal{I}_{k^{[-]}}, \tag{6}$$

with $\mathcal{I}_k = (0 \ \cdots \ 0 \ \underbrace{I}_{k^{\mathrm{th}} \text{ element}} \ 0 \ \cdots \ 0)$. Then, the update of a constraint turns into

$$\mathbf{x}^{t+1} \quad = \quad \mathbf{x}^t + \lambda |\mathcal{P}_{ji}| \mathbf{M}^{-1} \Omega'_{ji} r'_{ji}, \tag{7}$$

where $|\mathcal{P}_{ji}|$ refers to the number of nodes in $\mathcal{P}_{ji}$. In Eq. (7), we replaced the preconditioning matrix $\mathbf{H}^{-1}$ with its scaled approximation $\mathbf{M}^{-1}$ as described in [9]. This solution avoids a computationally expensive matrix inversion.

2.2. Incremental parameterization

When the network must be updated to account for the online addition of new constraints, in the batch version of the algorithm the optimization step involves the whole set of constraints, i.e. the new ones along with those already considered. However, adding new constraints often affects only a small portion of the nodes. Thus a complete optimization may be unnecessary and sometimes turns out even detrimental to the convergence to the correct map, since gradient descent changes also the configuration of already adjusted nodes.

An incremental variant of the previous algorithm has been proposed to make it suitable to solve online mapping problems [10]. The adaptation is obtained combining three different improvements to the base version of the algorithm.

- *Constraint Selection*. When adding a constraint $\langle j, i \rangle$ to the graph, a subset of nodes needs to be updated. Nodes involved in the update are identified by the minimal spanning subtree that contains nodes j and i (call $topNode(\langle j, i \rangle)$) the root of this tree). The subgraph to be optimized is detected with a variant of breadth-first search starting from $topNode(\langle j, i \rangle)$. The number of operations required for each optimization depends on graph topology: a large amount of time is saved when the robot closes small loops, whereas the technique may provide limited improvement with large cycles.

- *Adaptive Learning Rates*. Learning rate defines the magnitude of update in stochastic gradient descent. Thus a selective tuning of this parameter for each node limits adjustments to those parts of the network that are far from convergence. Learning rates values are initialized at the addition of a new constraint and then decrease at each iteration of the algorithm.

  A newly added constraint $\langle j, i \rangle$ introduces new information, in particular when a loop is closed. Thus the initial value of learning rate mainly depends on a term representing the information gain:

  $$\beta_{ji} = \Omega_{ji}(\Omega_{ji} + \Omega_{ji}^{\text{graph}})^{-1} \tag{8}$$

  where Ω_{ji} is the information matrix of the new constraint and $\Omega_{ji}^{\text{graph}}$ is an information matrix representing the uncertainty of the constraints in the network. The learning rate λ'_{ji} is chosen so that the value of the gradient is approximately $\beta_{ji}\, r_{ji}$. The value λ'_{ji} is then propagated to the nodes found with constraint selection.

  At each iteration learning rates are decreased according to a sufficiently smooth law, e.g. generalized harmonic progression.

- *Network Optimization Scheduling*. When a new constraint is added to the network, the global error is computed adding a new term to the previously computed value:

  $$e_{\text{new}} = \sum_{\langle j,i \rangle \in \mathcal{C}_{\text{old}}} r_{ji}^T \Omega_{ji} r_{ji} + \sum_{\langle j,i \rangle \in \mathcal{C}_{\text{new}}} r_{ji}^T \Omega_{ji} r_{ji}. \tag{9}$$

  To avoid unnecessary computations, we perform the optimization only if needed. This is the case when the newly incorporated information introduced a significant error compared to the previous error of the network. Several heuristic rules can be adopted to assess such a condition, and the following one has been chosen in this work:

  $$e_{\text{new}} - e_{\text{old}} > \alpha \max_{\langle j,i \rangle \in \mathcal{C}_{\text{old}}} r_{ji}^T \Omega_{ji} r_{ji} \tag{10}$$

  where α is a factor that measures how many "bad" constraints are acceptable before optimizing the network. Note that Eq. (10) is slightly different from the rule proposed in [10].

3. BUILDING THE CONSTRAINT NETWORK FROM LASER SCANS

In this section we describe the part of the system that transforms raw sensor data into a network of constraints ready to be processed by the optimizer described in the previous section. The aim is to extract from laser scans a graph consisting of poses and constraints among pairs of poses. In order to compute the relationship between two poses, an observation is anchored to a pose. Therefore, environment features are not explicitly modeled and a global map is defined by a collection of poses like in [3]. The scan associated to each pose provides a sort of local map. A more effective way to build a map patch would consist in accumulating several consecutive scans into a synthetic representation, e.g. a grid map. In this work, however, we adopted the straightforward solution, as we focused on the integration of TORO with laser scan processing.

Thus, the network builder previously sketched requires two basic operations. First, transformation from pose j to pose i must be recovered. This can be done by matching laser scans which can be overlapped for a consistent portion. Then, selection of matching candidates allowing loop closure must be made. Figure 1 displays the main steps of the algorithm. Initially, the current scan is compared with the immediately previous one to recover the intial guess of the corresponding robot pose. Afterwards, the algorithm selects the candidate scans for data association, extracts the constraints with the current pose, and performs an optmization step on them. Thus, the network builder previously sketched requires two basic operations. In the following, the two basic operations are discussed thoroughly.

3.1. Computation of pose constraints

A constraint between two poses consists of a relative transformation vector (translation and rotation) and second order statistics, i.e. the information matrix describing the uncertainty on the transformation. The primitive operation to extract these data from a pair of laser scans is a scan matching algorithm. Several scan matching algorithms have been proposed and many of them belong to the *iterative closest point* (ICP) approach. In this work, we adopted the ICP variant described in [13], whose implementation is also available.

Scan matching provides an estimation of the transformation that overlaps the common part of two scans, but constraint description requires second order statistics. The adopted scan matcher provides also an estimation of the covariance based on error function minimization [13].

3.2. Data association

In SLAM and graphical models in particular, there are two kinds of data association, as remarked in [4]. The robot moves along a path defining a natural order and association among consecutive poses. Thus a new pose has a link to the previous pose based on odometry, and to several of the

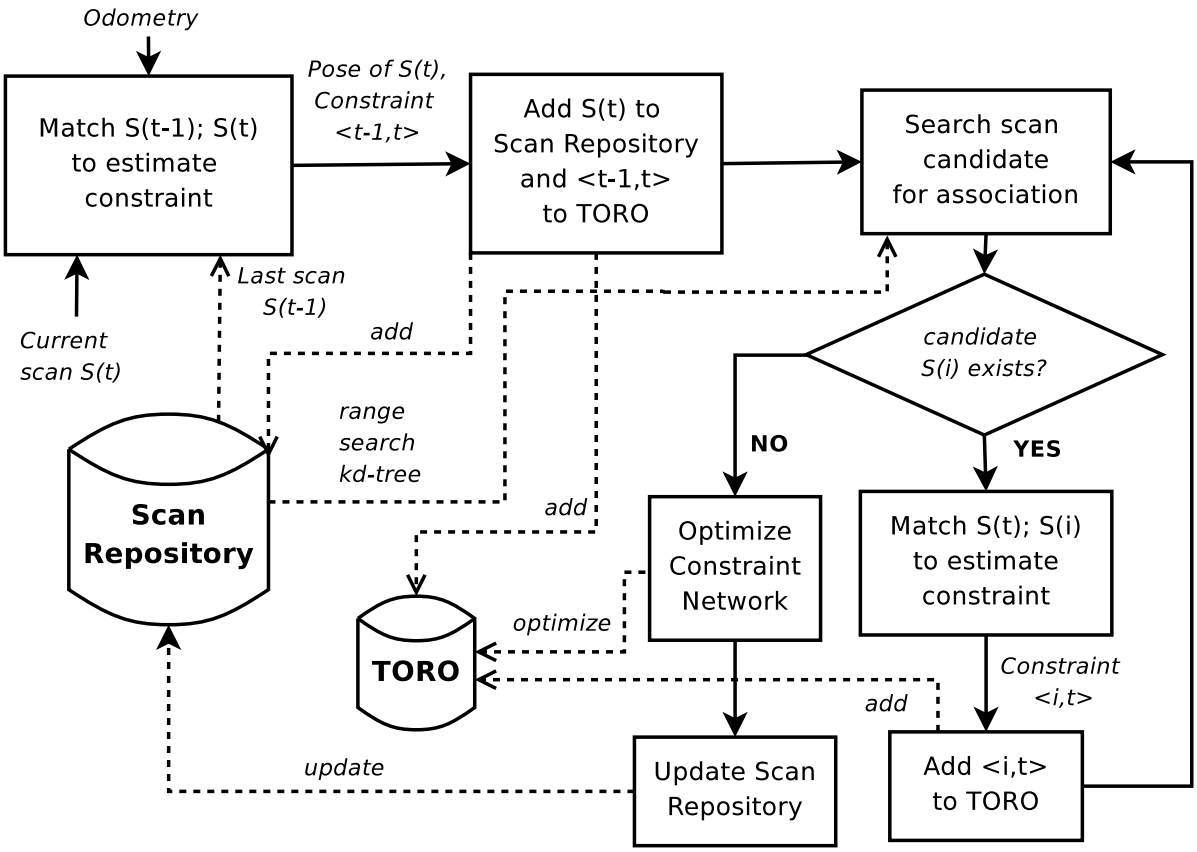

Figure 1: General schema of the algorithm building the constraint network.

previous poses based on scan overlaps and the resultant scan matches.

When the robot reaches an already explored area, the current pose matches with a portion of map too. These correspondences between poses introduce loops into the graph and allow map adjustment. They are also critical for the solution of the mapping problem because wrong correspondences lead to incorrect solution. In offline graphical approaches to SLAM an optimal match is found since data association is performed when all data are available. Since our algorithm is incremental, wrong associations cannot be undone, similarly to other online SLAM algorithm.

Loop closure is a difficult problem compared to the local match of scans. First it requires global search of poses already stored in the map and this operation should be performed online. In this paper, links between current pose and candidate poses are tested with scan matching as previously illustrated. However, match of non-sequential scans is difficult because of the loose overlap of scans with poor estimation of relative pose. Furthermore, matching candidates are often redundant: possible high concentrations of poses in specific areas of the map depend on the speed of the robot collecting sensor data. When the system tests loop closure candidates, comparison with noisy overlapping scans may produce inconsistent constraints and is computationally expensive as well.

In the work reported in this paper, we solve the data association problem by means of a straightforward algorithm that combines several simple solutions. A first requirement is to avoid overpopulated maps. Since each scan virtually corresponds to a local map that could increase the noise and contribute to the complexity of association, a filtering criterion is required. In the current system, a new pose is added to the map only if its distance from the previously inserted pose exceeds a predefined threshold, where the distance metric also considers pose orientation. This rule, along with other features, has been chosen to favor simplicity and efficiency in view of a real-time implementation. An alternative representation of the environment, such as a collection of local grid submaps accumulating multiple scans, would allow more robust data association techniques, at the expense of some additional computation. At a higher level, the adopted tree network optimizer is able to handle node equivalences [12] and to collapse two poses in a single node.

The initial value of a newly added pose in TORO depends on the constraint that links the new node with a previously added one. Thus, the first constraint can be recovered using odometry as initial guess and correcting the value by a scan alignment with the last inserted scan. The core of the map builder is therefore the loop closure detector that performs in two steps. After the addition of a new pose, can-

didate cycles are found via range search on a *kd-tree* [14]. Of course, recent scans are not considered since they are directly linked through the robot path sequence. The size of the searching region depends on the size of current map and on the uncertainty in the estimation. Furthermore, if a loop has been closed recently, the search is concentrated on a smaller area. Then the list of candidates is reduced performing the usual matching test. Rejection of constraints relies on multiple criteria, including the match covariance values, the distance between the pair of poses, and the extent of the overlapping area. When two matching nodes refer to the same location, the situation is handled by TORO with node equivalence.

The main drawback of the adopted approach is the need to choose several parameters like the range size for the preliminary search and the threshold for covariance. These parameters require a careful tuning which is currently performed by experimentation. An example of critical parameter is reported in the next section.

4. RESULTS

In this section, we evaluate the effectiveness of the proposed system for map learning based on the tree network optimizer. To test our system with real robot data, we used the freely available ACES dataset [15]. This dataset provides a sequence of laser scans acquired sequentially as the robot moves in an environment. The reported experiment consists in the extraction of constraints relating consecutive poses from pairs of scans and in the detection of cycles with the previously discussed technique.

In section 3, we mentioned the importance of carefully tuning several parameters of the algorithm. In this experiment, the minimum distance between two consecutive poses in the map has been set to 20 cm. The maximum size of search area where candidate associations are detected has been chosen proportional to the size of already explored map by factor 0.1. This range suddenly decreases when a loop is found and then increases progressively.

Figure 2 represents a typical loop detection situation. The top map is built using odometric information and pairwise matching of consecutive scans. A duplicate of the top of the hallway, which has been already visited, is clearly visible. The proposed algorithm allows the two frames depicting the same place to be linked together, as shown in the bottom map. Figure 2 also illustrates the current limitation of the algorithm: scan matching recovers topological correctness, but it provides limited accuracy in the estimation of the transformation between two distant frames. Thus, a simple map model can provide the structure of the network, whereas a more sophisticated matching algorithm, processing map patches, might be required to improve the metric accuracy of the map.

Figure 3 shows the global error trend with the ACES dataset. The error value grows approximately linearly with the number of constraints added to the optimizer (note the logarithmic scale), although a few spikes are apparent. These spikes are due to the constraint selection illustrated in section 2 and are particularly prominent when a large loop is closed. Thus, the proposed algorithm for data association does not degrade performance with respect to the results reported in [10].

Figure 2: Map of the environment of dataset ACES before (top) and after (bottom) a loop closure is solved with the proposed algorithm.

5. CONCLUSION

In this paper, we have presented a map estimator that builds a constraint network from the observations and solves constraints using an incremental tree network optimizer. The algorithm accepts odometric information and laser scans as an input and produces a representation of the environment

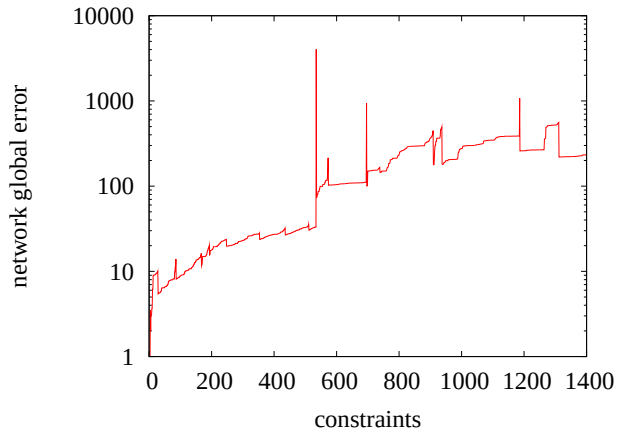

Figure 3: Global error of the constraint network based on the number of constraints.

in the form of a collection of adjusted scans. This simple kind of map has the advantage of being easy to implement and to manage, even though in our current implementation it suffers from limited accuracy when large cycles are closed. In our future work, we expect to achieve better accuracy by integrating several scans into local patches.

6. ACKNOWLEDGMENTS

We thank Giorgio Grisetti for providing us the incremental TORO and Andrea Censi for making available his scan matcher. The first author is also grateful to Cyrill Stachniss, Edwin Olson and Wolfram Burgard for the discussions on stochastic gradient descent techniques.

7. REFERENCES

[1] J. Leonard and H. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1991.

[2] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[3] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Journal of Autonomous Robots*, vol. 4, 1997.

[4] 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, pp. 318–325.

[5] T. Duckett, S. Marsland, and J. Shapiro, "Fast, on-line learning of globally consistent maps," *Journal of Autonomous Robots*, vol. 12, no. 3, pp. 287 – 300, 2002.

[6] U. Frese, P. Larsson, and T. Duckett, "A multilevel relaxation algorithm for simultaneous localisation and mapping," *IEEE Transactions on Robotics*, vol. 21, no. 2, pp. 1–12, 2005.

[7] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Fast incremental smoothing and mapping with efficient data association," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2007.

[8] U. Frese, "Treemap: An $o(logn)$ algorithm for indoor simultaneous localization and mapping," *Journal of Autonomous Robots*, vol. 21, no. 2, pp. 103–122, 2006. [Online]. Available: http://www.informatik.uni-bremen.de/ ufrese/published/fresear06b.pdf

[9] 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)*, 2006, pp. 2262–2269.

[10] 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)*, 2008.

[11] C. Estrada, J. Neira, and J. Tardós, "Hierachical slam: Real-time accurate mapping of large environments," *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 588–596, 2005.

[12] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard, "A tree parameterization for efficiently computing maximum likelihood maps using gradient descent," in *Proc. of Robotics: Science and Systems (RSS)*, Atlanta, GA, USA, 2007. [Online]. Available: http://www.informatik.uni-freiburg.de/ stachnis/pdf/grisetti07rss.pdf

[13] A. Censi, "An accurate closed-form estimate of ICP's covariance," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2007. [Online]. Available: http://purl.org/censi/2006/icpcov

[14] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Com. ACM*, vol. 18, no. 9, pp. 509–517, September 1975. [Online]. Available: http://portal.acm.org/citation.cfm?id=361007

[15] A. Howard and N. Roy, "Radish: The robotics data set repository, standard data sets for the robotics community." [Online]. Available: http://radish.sourceforge.net/