METRIC-TOPOLOGICAL MAPS FROM LASER SCANS ADJUSTED WITH INCREMENTAL TREE NETWORK OPTIMIZER

Dario Lodi Rizzini, Stefano Caselli

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 two map builders that process laser scans in order to extract the constraint network for the optimization algorithm. The first algorithm builds a map in the form of a collection of scans corresponding to a subset of the poses of a robot moving in the environment. Even though such a solution has the advantage of simplicity, it requires careful processing of data associations. After a preliminary selection of pose constraints candidates, relative pose is computed through standard scan matching techniques. The second map builder stores a hybrid metric-topological representation: the map consists of a graph whose nodes contain local occupancy grid maps and whose edges are labeled with relative pose between pairs of nodes. Each patch map summarizes the information of consecutive raw scans and such a richer representation better solves loop closure. Association between pairs of local maps is then performed and tested using correlation based techniques. Our aim is to illustrate the effectiveness of a tree network optimizer integrated with simple methods for data association. Experiments reported in the paper show that a compact system integrating the optimizer and one of two versions of the map builder works reasonably well with commonly used benchmarks.

Key words: Mapping, Maximum Likelihood

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 the 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].

One drawback of the above algorithms aiming to solve graph-based SLAM formulations is that they 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 straight-forward 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. Two versions of the constraint network builder have been developed. They both take odometric information and laser scans in input and use an hybrid metric-topological map, i.e. a map consisting of a graph whose nodes are local metric maps and whose edges store topological relationships between pairs of local maps. However, the two map builders differ in their representation of the local map and method for data association.

The first version maintains local maps in the form of raw laser scans and each scan corresponds to the robot pose from which the scan has been acquired. While easy to implement, this solution requires a more careful procedure in detecting large cycles. A single scan often does not provide a sufficient description of a region of the map and data association is difficult. The implemented map builder correctly recovers the topology of the map, but it does not estimate relative position and orientation between two scans with adequate accuracy.

The second constraint network estimator builds a local patch exploiting several consecutive sensor observations as suggested in [4, 11]. The local map is an *occupancy grid map* and the cells of the grid are filled according to multiple scans. Such a thorough representation allows better results when a loop is closed: the relative location between two submaps is recovered by using correlation based techniques and then an acceptance test is performed by counting cells with agreeing values. A careful detection of the relative orientation between grid maps is required before each comparison. Such a condition is ensured by processing the Hough spectrum of patches.

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

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

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:

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

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}:

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

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:

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

where parent(i) refers to the index of parent of node i. In the original formulation [9] 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

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

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

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

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

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

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. Eq. (10) is slightly different from the rule originally 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. Therefore, map estimation does not depend explicitly on environment features, but only on robot poses like in [3].

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.

However, scan matching is not enough to build the constraints network. This technique is effective when two scans significantly overlap and the initial guess of relative pose is not too far from the correct value. These conditions hold when consecutive poses are compared, but the comparison fails when

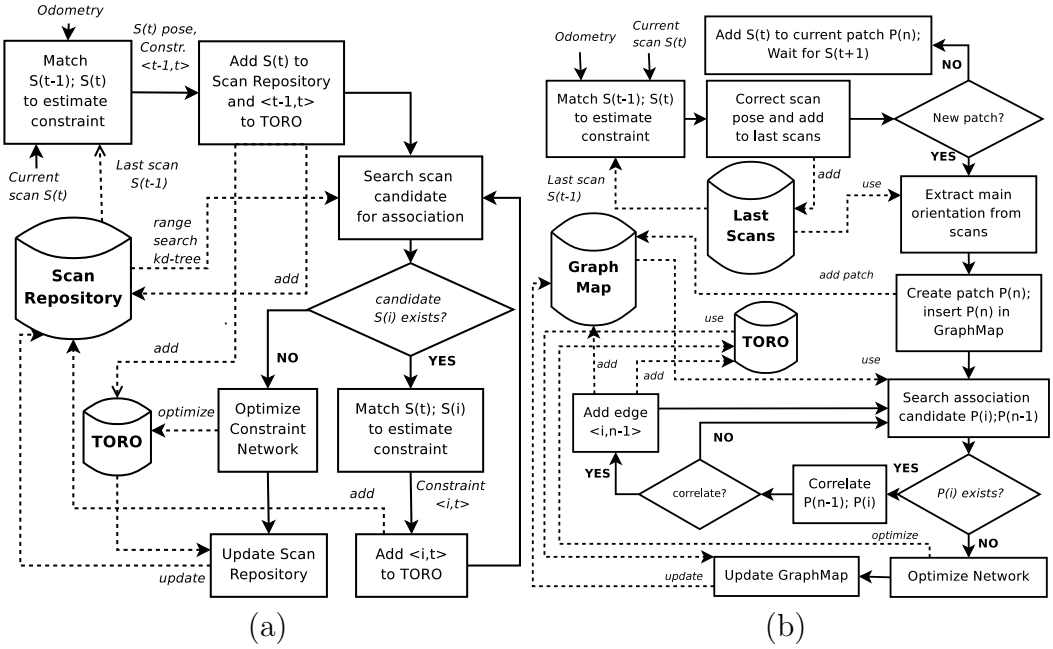

Figure 1: General schemas of the two algorithms building the constraint network: (a) `ScanMap` (b) `GraphMap`.

the robot moves in a previously explored region and tries to match current observation with the stored map. Two different network builders, named `ScanMap` and `GraphMap` respectively, have been developed.

`ScanMap` exploits a straightforward representation of the map: each node of the graph stores a pose and the corresponding laser scan acquired when the robot visited the pose. The scan associated to a location provides a sort of local map. Figure 1(a) displays the main steps of the algorithm. Initially, the pose of new node is initialized with the relative motion refined by scan matcher. When a loop is closed, the algorithm selects candidate scans for data association, tries to match the current pose and candidates and performs an optimization step on them. Unfortunately, constraints extracted with this procedure are not always estimated correctly, as will be shown later.

`GraphMap` (figure 1(b)) has been developed to overcome `ScanMap` limitations and is similar to the method described in [4]. In particular, the local map stored in each node is an occupancy grid map obtained from several aligned consecutive scans. Thus, the number of nodes significantly decreases and loop closure becomes more effective since each local map patch better

describes a location. Data association is performed with a correlation-like algorithm. In the following, the two algorithms are discussed thoroughly.

3.1. ScanMap

`ScanMap` represents the map as a collection of pairs of poses and scans. Such solution is quite straightforward and completely relies on scan matching technique for both pose initialization and loop closure. The initial value of a pose depends on the value of the previous pose and on the relative transformation between them due to the motion of the robot. Relative pose is first estimated with odometry and then is corrected by scan alignment.

The core of the map builder is therefore the loop closure detector that performs in two steps. After the addition of a new pose, candidate cycles are found via range search on a *kd-tree* [14]. The size of the searching region depends on the size of the current map and on the uncertainty in estimation. Furthermore, if a loop has been closed recently, search is concentrated on a smaller area.

Candidate poses for data association are then selected from the poses inside the searching area. Recent nodes, i.e. nodes directly connected to the current node through a path that falls inside the searching area, are not considered to avoid too small or repeated loops. Several criteria are applied to reduce candidates and detect correct associations. Alignment of the current scan with a candidate scan estimates their relative pose. Scan matching is performed starting from null initial guess. The most likely candidate is then selected by evaluating the match covariance values, the distance between the pair of poses, and the extent of the overlapping area.

Main drawbacks of the approach are the limited metric accuracy and the need to choose parameters for heuristic rules like the range size for the preliminary search and the threshold for covariance. `ScanMap` will be further discussed in the next section.

3.2. GraphMap

`GraphMap` consists of a graph whose nodes store poses with a local map and whose edges correspond to constraints between pairs of nodes like in `ScanMap`. The difference between the two algorithms lies in the interpretation of nodes. Indeed, the pose stored in each node is the local reference frame of the local map and does not correspond to the robot trajectory. Therefore, a local map consists of an occupancy grid map built from a set of consecutive laser scans. Each cell of the local map has one of the following values: *empty*,

occupied or *unknown*. A laser scan is inserted in a map patch by placing the center of the scan in the map and tracing each beam until the measured distance is reached. Traversed cells are filled with *empty* value, until the last cell is labeled as *occupied*. Such a representation, also called *map patch*, is more expressive than a collection of raw scans and allows a more effective data association.

Figure 1(b) illustrates the main steps of the algorithm. First, the new scan is aligned with the previous one by scan matching to improve the estimation of relative motion provided by odometry. Then, the observation is stored in a temporary buffer containing the last scans available for any comparison. If the distance between the current robot pose and the center of the map patch is less than a given threshold, then the scan is used to fill the current grid map. For the experiments illustrated in the next section the threshold was set to 6 m. Otherwise a new patch is created.

Since occupancy grid maps are suitable for the representation of obstacles extending along directions orthogonal to patch borders, the algorithm extracts the main orientation of the first scan added to the patch. The main orientation intuitively corresponds to the bundle of parallel lines that better fits the length of the obstacles and is operatively computed by searching maxima in the *Hough spectrum* [15]. Finding a principal direction also helps when two grid maps are compared for data association. Main orientation is usually well-defined for indoor structured environment.

After the insertion of a new node, `GraphMap` searches for candidate nodes for data association with the just completed map patch. Candidates are found with range search as illustrated before and then are matched with the current patch by exploiting a correlation-like function. Assuming that relative orientation of two local maps is obtained by comparing their main orientations, two occupancy grids are better processed by a correlation function that operates on pairs of matching cells. In order to limit computational load, displacement between two patches is computed by correlating their separate histograms for axes x and y. A histogram bin counts the number of occupied cells whose projection falls inside the bin. Displacement between two maps is computed searching for maxima of correlations between x-histograms and y-histograms. Finally, the algorithm performs an acceptance test on the overlapping parts of the two patches placed according to the estimated relative position and orientation. This test counts the number of cells with equal values and ignores the cells with *unknown* value. If the outcome of the test is positive, i.e. the number of agreeing cells is larger than a given threshold,

a link between the two nodes is added to `GraphMap` and to TORO and an optimization is performed.

4. Results

In this section, we evaluate and compare the effectiveness of the two proposed systems for map learning based on the tree network optimizer. To test the two map builders with real robot data, we used the freely available ACES dataset [16].

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 techniques.

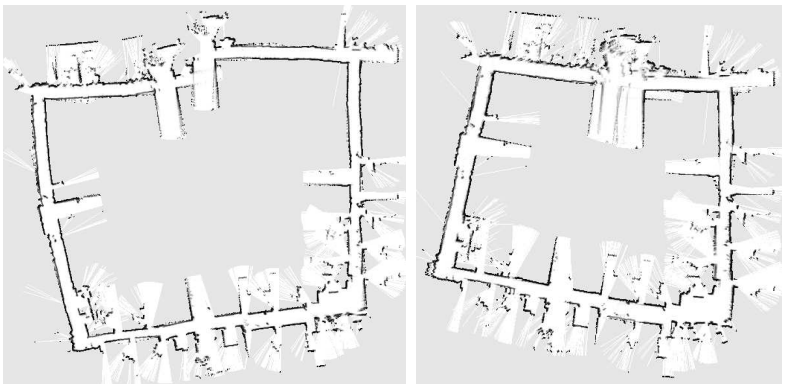

Figure 2: Map of the environment of dataset ACES before (left) and after (right) a loop closure is solved with `ScanMap`.

Figure 2 illustrates the effectiveness, but also the main problem with the first map estimator proposed in this paper: when a loop closure is performed, `ScanMap` correctly recognizes a known region, but the scan matcher does not provide a good estimation of relative constraint between the scans stored in matching nodes. The result shown in Figure 2(right) shows the effect of the new constraint: the estimated distance between the two places is too short and the resulting map is pulled on the top part.

The map estimation performed by `GraphMap` is shown in figure 3. The internal representation adopted by the algorithm is apparent: the map consists of a set of occupancy grid maps associated to the nodes of the network. The boundaries of each map are represented by the rectangular bounding

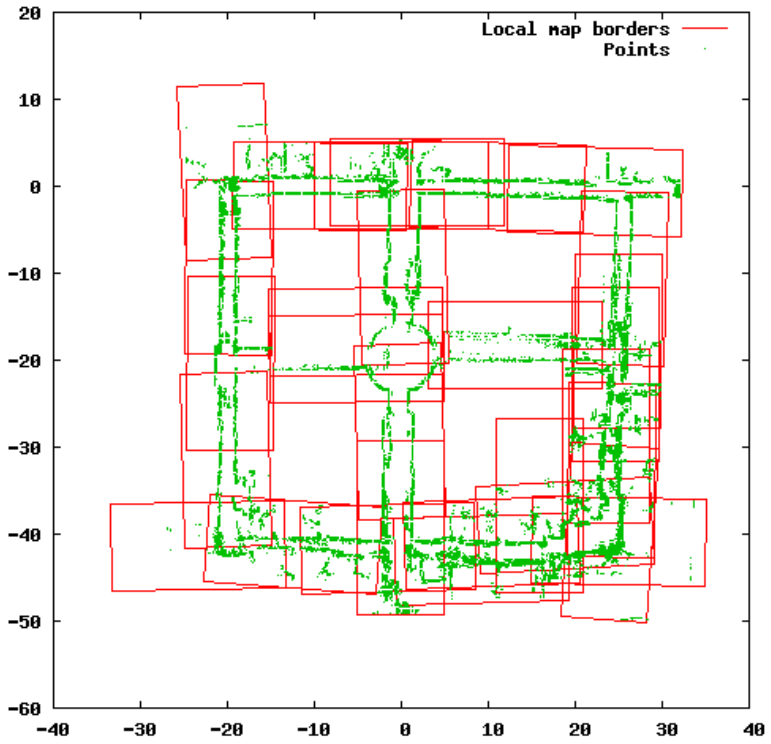

Figure 3: Map of the environment of dataset ACES after an optimization performed with `GraphMap`.

box and the *occupied* cells are represented by a point. For readability, the output does not distinguish *empty* cells from *unknown* ones. Note that the orientation of each submap is consistent with the orientation of map walls of the represented region.

The size and resolution of each grid map are fixed. In this experiment the size of the grid map is set to 20.0×10.0 m with a resolution of 0.2 m. The size of the map patch may appear large if compared to the size of the whole map (about 55×45 m), but it is a trade-off between the local adjustment of consecutive scans allowed by the scan matcher and the need for distinguishable map. Indeed, smaller map patches appear too similar to each other and the resulting loop closure detection is too prone to matching errors.

The execution of `GraphMap` with online optimization using TORO on an Intel Core 2 Quad Q9300 2.5 GHz requires about 42.729 s. The mean time required to create a single map patch and to perform optimization step is

$0.029s$. Neither the graph builder nor network optimizer have a significant impact on the total amount of time. The most computationally expensive operation is the alignment of 7169 scans with the scan matcher. `GraphMap` generates only 41 nodes compared to the about 700 nodes used by `ScanMap`. Thus, `GraphMap` reduces the complexity of the network optimization phase.

5. Conclusion

In this paper, we have presented two map estimators that build a constraint network from laser scans and solve constraints using an incremental tree network optimizer. The first map builder generates a map in the form of a collection of scans corresponding to a subset of the poses of a robot moving in the environment. The second map builder adopts a hybrid metric-topological representation: a map consists of a graph whose nodes contain local occupancy grid maps and whose edges are associated to relative poses between pairs of nodes. Both algorithms accept odometric information and laser scans as input, but the latter has demonstrated better performance in loop closure detection. Furthermore, `GraphMap` largely reduces the number of poses stored in the graph since scans are merged into an occupancy grid map. The two map builders have been assessed along with the incremental tree network optimizer using the classic ACES dataset.

6. Acknowledgments

We thank Giorgio Grisetti for providing us the incremental TORO and Andrea Censi for making available his scan matcher. We are also grateful to Nicola Bova and Giorgia La Rosa for their help in the development of the map estimator.

This research has been supported by LARER and AER-TECH initiatives of Regione Emilia-Romagna, Italy.

References

[1] J. Leonard and H. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," *IEEE Transactions on Robotics and Automation*, vol. 7, pp. 376–382, 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, no. 4, pp. 333–349, 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, pp. 1670–1677.

[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.

[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, pp. 1880–1885.

[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.

[13] A. Censi, "An accurate closed-form estimate of ICP's covariance," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2007.

[14] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Com. ACM*, vol. 18, no. 9, pp. 509–517, September 1975.

[15] S. Carpin, "Merging maps via Hough transform," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2008.

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