Go straight, Turn Right: Pose Graph Reduction through Trajectory Segmentation using Line Segments

Yasir Latif and José Neira

Abstract— **With better hardware and more efficient graph-SLAM solvers, we are able to solve increasingly large mapping problems. An actual implementation of a mapping problem as a pose graph requires a certain amount of discretization of the information coming from odometry. Such discritizations are either sensor dependent or use a minimum distance travelled heuristic to add poses to the graph. In this work, we explore the question: how much information we can discard and still be able to get a correct map estimate using the pose graph formulation. We approximate the robot trajectory by a sequence of lines leading to a reduced representation of the original pose graph. This reduction is carried out by using an incremental algorithm that adds new poses to the reduced graph when the perpendicular distance for the current estimated line exceeds a threshold. The reduced representation allows us to recover a part of (or the full) graph when needed. This is achieved by exposing the reduced graph to the optimizer but at the same time not discarding the original pose graph. We show the application of our proposed method on real world datasets and illustrate the accuracy and efficiency with which a reduced representation can approximate the original pose graph problem.**

I. Introduction

Recently, the graph based formulation to solve the SLAM problem has become a common choice. Even though this representation was proposed a long time ago [1], it has made a comeback thanks to advances which allow efficient solutions for the non-linear optimization problem. Most of the current optimization back-ends take advantage of the sparse nature of the SLAM problem, for instance iSAM [2], HOG-Man [3], and g2o [4].

Graph-SLAM is traditionally divided into two separate parts: (a) the front-end which extracts constraints from the sensor input and adds them to the pose graph as edges, and (b) the back-end which is responsible for calculating the most likely configuration of the nodes, given these constraints. The front-end, based on the sensory input, has to decide when to add new constraints to the pose graph. Traditionally it is done after a certain amount of time has elapsed, such as for a laser scanner working at a fixed frequency, or when there is significant change in the robot pose or environment, as is the case in key-frame based approaches [5], [6].

As the robot explores the environment, the pose graph representing the SLAM problem grows with the exploration time. The number of links in the graph, as well as the graph structure, determines the computational complexity of the optimization process. In recent literature, various methods have been proposed to deal with the increasing number of nodes in the graph. Grisetti et al. [3] represent the pose graph at different levels in a hierarchy, where the lowest level represents the full pose graph and successive higher levels represent sampled versions of the lower levels. Updated are propagated between different levels when needed. Ila et al. [7] use an information theoretic method for selecting the constraints that should be added to the pose graph. Loop closures are added by assessing the possible information gain. This provides a sampled version of the pose graph but the graph still grows with exploration time.

More recently, approaches have been proposed that allow the pose graph to grow with the area explored as opposed to the exploration time. Kretzschmar et al. [8] presented an information theoretic pruning for laser-based maps. The main idea is to consider the mapped area as a grid-map and prune laser scans based on an information gain measure. For Visual SLAM, Johannsson et al. [9] showed an approach that scales with the explored area by adding constraints during revisits to the existing pose graph rather than adding new vertices in the graph. They show results for a large stereo sequence map spanning a time of nine hours. Walcott-Bryant et al. [10] presented a mapping solution namely DG-SLAM for low dynamic environments in which out-dated information is pruned based on new observation, thus maintaining a reduced map that changes with time.

Some approaches approximate the marginal covariance in

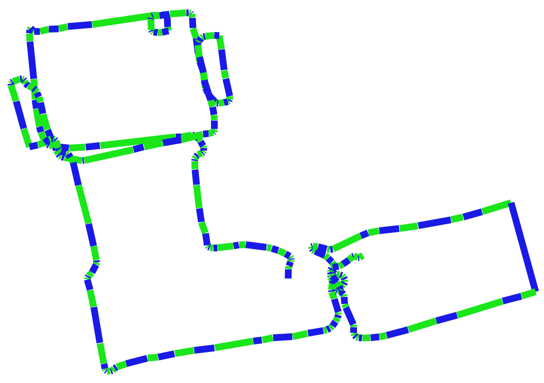

Fig. 1. Odometry from the Bicocca dataset segmented with a threshold of 0.05m. Alternating colours represent different segments. The reduced representation contains 1127 nodes compared to the original 8358 nodes.

Yasir Latif and José Neira are with the Instituto de Investigación en Ingeniería de Aragón (I3A), Universidad de Zaragoza, Zaragoza 50018, Spain. {ylatif, jneira}@unizar.es.

This research has been funded by the Dirección General de Investigación of Spain under projects DPI2009-13710 and DPI2012-36070 and by DGA-FSE(group T04).

Algorithm 1 Incremental Line Estimator

```
Input: $poses(x_{start} : x_k)$, $start$, $end$, $k$, $threshold$, $segmentList$
Output: $start$,$end$, $segmentList$
  if $poses.size() == 2$ then
    start $\leftarrow x_0$
    end $\leftarrow x_1$
  else
    $L = line(x_{start}, x_{end})$
    $i = \text{argmax}_p\, distance(x_p, L) \bigwedge p \in (start, k)$
    if $distance(x_i, L) > threshold$ then
      $segmentList \leftarrow segmentList \cup segment(start, i)$
      $start \leftarrow i + 1$
      $end \leftarrow k$
    else
      $end \leftarrow k$
    end if
  end if
```

order to identify which nodes to eliminate. In this context Kretzschmar et al. [11] presented a Chow-Liu tree based approximation for the marginal covariance. Similarly, Carlevaris-Bianco and Eustice [12] show a method for marginalization over cliques of nodes which can later be sparsified using a Chow-Liu tree approximation.

In this work, we take a very simple approach to the problem of a reduced pose graph representation. We reduce the pose graph taking into account the underlying trajectory of the robot. We employ an incremental variant of the well-know split and merge algorithm [13] for this estimation, that only utilizes the "split" part of the algorithm as the trajectory is incrementally approximated by lines segments therefore line segments are only split but never merged. The main idea is to keep the computational overhead small by reducing the number of poses that represent the mapping problem. In addition, we want to be able to recover the full pose graph once we have optimized a reduced version of it. Moreover, we are interested in finding out how such a simple approach effects the final estimate in terms of accuracy and the time needed to compute the optimal pose estimates.

In the next section we summarize the formulation of the SLAM problem as a pose graph. In section III we provide the detail of our method. Experiments are detailed in section IV along with in-depth evaluations of the effect of reduction on different aspect of the graph-SLAM problem. Finally, in section V further discussion and conclusions about this work are presented.

II. The Pose Graph Formulation

The graph based formulation for SLAM, the so-called "graph-SLAM", models robot poses as state variables in the graph's nodes and constraints as factors on the graph's edges. The factors represent a distance to minimize between the poses and the observations given by the sensors. In the Gaussian assumption case, sensor noise is modelled using the covariance or the information matrix. Let $\mathbf{x} = (x_1 \ldots x_n)^T$ be a vector of parameters that describes the configuration of the nodes. Let t_{ij} and Ω_{ij} be the mean and the information matrix of the observation of node j from node i. Given the

Algorithm 2 Compose

```
Input: $poses(x_i : x_j)$, $i$, $j$
Output: $t_{ij}$,$\Sigma_{ij}$
  $k \leftarrow i + 1$
  while $k < j - 1$ do
    $t_{ik} \leftarrow t_{i,k-1} \oplus t_{k,k+1}$
    $\Sigma_{ik} \leftarrow J_1 \Sigma_{i,k-1} J_1^T + J_2 \Sigma_{k,k+1} J_2^T$
    $k \leftarrow k + 1$
  end while
```

state $\mathbf{x}$, let the function $f_{ij}(\mathbf{x})$ be a function that calculates the perfect observation according to the current state. The residual r_{ij} can then be calculated as:

$$r_{ij}(\mathbf{x}) = t_{ij} - f_{ij}(\mathbf{x}) \tag{1}$$

Constraints can either be introduced by odometry which are sequential constraints $(j = i + 1)$, or from place recognition system which are non-sequential. The amount of error introduced by each constraint, weighed by its information, can be calculated as:

$$d_{ij}(\mathbf{x})^2 = r_{ij}(\mathbf{x})^T \Omega_{ij} r_{ij}(\mathbf{x}) \tag{2}$$

and therefore the overall error, assuming all the constraints to be independent, is given by:

$$D^2(\mathbf{x}) = \sum d_{ij}(\mathbf{x})^2 = \sum r_{ij}(\mathbf{x})^T \Omega_{ij} r_{ij}(\mathbf{x}) \tag{3}$$

where $d_{ij}(\mathbf{x})^2$ is the pairwise factor of the present variables in nodes i and j. The solution to graph-SLAM problem is to find a state $\mathbf{x}^*$ that minimizes the overall error.

$$\mathbf{x}^* = \underset{\mathbf{x}}{\text{argmin}} \sum r_{ij}(\mathbf{x})^T \Omega_{ij} r_{ij}(\mathbf{x}) \tag{4}$$

III. Method

In the proposed method, we exploit the fact that robots and vehicles tend to move along straight lines. Moving along a straight line signifies that the robot's translation parameters change but the orientation parameter varies slightly or remains the same. The SLAM problem is non-linear due to the angular component of each constraint. If the robot travels in a nearly straight line, due to the small angle approximation, the constraints contribute linearly towards the solution. We can therefore replace all such constraints with a single constraint, in effect reducing the number of poses representing the mapping problem.

Given a pose graph, we can efficiently estimate a line based approximation for the robot's trajectory. Line approximations have been used previously for fitting line segments to 2D range scans [13]. As with any approximation method, the user has to decide what threshold to set. In our case, this threshold represents the maximum point-line distance after which the point is considered to not belong to the line. In other words, all the points should have a distance lower than this threshold from the line-segments by which they are approximated.

The process of line-fitting is given in Algorithm 1. Given a new pose (k), we have to decide if it agrees with the line-segment being estimated (which is represented by the $start$ and end vertex labels) or should the line be split into two segments. This is done by calculating the distances from the current line to all the points between the start and the current node (k), and finding the node (i) which lies the farthest away from this line. If this distance is greater than the specified threshold, the line is terminated at node (i) and the new line starts at the node next to it ($i{+}1$), with the current node (k) as its end point. When the line terminates, the poses from $start$ to end are the nodes in the graph that lie along a straight line. These labels are the end-points of the line segment. The reduced graph contains only the corresponding two nodes per segment and the computed transformation.

This algorithm assumes that the nodes are labelled along time such that the first node to arrive in the graph has a label zero and each new node introduced increases this label by one.

The pose graph until now is assumed to consist of just sequential constraints. An example output of the algorithm is shown in Fig. 1 where each new segment is represented with an alternating colour. The threshold used for this example is 0.05 meters.

We now address how loop closures can be incorporated in the reduced pose graph. Loop closures provide non-sequential constraints on the graph that help reduce the overall uncertainty and allow for a better map estimate. In this work, we assume that all the loop closures present in the graph are correct and we do not address the issue of incorrect loop closure that may arise due to perceptual aliasing (see Section V for discussion). The method described in this work keeps all the loop closing edges that are present in the graph. The information provided by loop closure is too valuable to be discarded.

We assume that the non-sequential edges encode loop closures and have a consistent sense of direction i.e loop closure are given by measurements between two non-consecutive nodes x_k and x_l where $k > l$. For each of the poses x_k and x_l not already present in the reduced graph, the segment to which the node belongs are retrieved. The corresponding segment which starts at node x_i and ends at x_j must satisfy $i < l < j$. The segment can now be split at x_l in to two segments (x_i, x_l) and (x_l, x_j). The constraints introduced by the front end are added to the reduced-graph without any modification. Incorporating loop closures information in the reduced pose graph is only possible if we do not discard the actual information that was introduced by the front end, which is precisely what our method does. We do not throw anything away, but instead expose a selected subset of the complete information to the back-end optimizer. This reduces both the number of vertices and the number of edges in the graph being optimized while allowing us to "on-demand" recreate the complete graph when needed, as explained later.

The output of the above process is a sorted list of vertices that belong to the reduced graph and are contained in the $segmentList$. The members of $segmentList$ are of the form $[(0,k),(k+1,j),(j,l),(l,m),(m+1,\dots]$. Disjoint segments are the segments that arise from sequential constraint, such as the first two in the list. They are disjoint because they have no nodes in common. Segments that are created due to loop-closings have overlapping end-points (such as the second and third pair in the previous list). We keep segments arising for sequential constraints disjoint in order incorporate one extra constraint (in this case $t_{k,k+1}$), allowing the two lines to "move" with respect to each other. Each element in this list is then passed onto Algorithm 2 which composes the corresponding constraints from the original graph for each segment and introduces this new constraint into the reduced graph using a first order approximation as explained below:

Given two constraints t_{ij} and t_{jk} with associated covariances Σ_{ij} and Σ_{jk}, the first order approximation of the resulting constraints t_{ik} is given by

$$t_{ik} = t_{ij} \oplus t_{jk}$$

and the corresponding covariance matrix Σ_{ik} by

$$\Sigma_{ik} \simeq J_1 \Sigma_{ij} J_1^T + J_2 \Sigma_{jk} J_2^T$$

where J_1 and J_2 are Jacobian of the composition function w.r.t t_{ij} and t_{jk} respectively.

Algorithm 2. calculates a chain of such transformation

$$t_{ij} = t_{i,i+1} \oplus t_{i,i+1} \oplus \dots \oplus t_{j-1,j}$$

along with the corresponding covariance matrix and adds the resulting constraint to the reduced pose graph.

Additionally, constrains between the end and start of consecutive segments (k and $k+1$ in the previous example) are also added to the reduced graph, along with all the loop-closing constraints.

IV. Experiments

In order to evaluate our method, we work with real datasets. The experiments shown here use the Bicocca dataset and the Bicocca Multi-session dataset from the RAWSEEDS project [14] as well as the New College dataset [15]. The experiments investigate how trajectory segmentation effects the translation error, uncertainty estimates, optimization time, and the number of vertices in the graph. All the experiments provide results with the full-graph as the reference.

The Bicocca and Bicocca-multisession datasets consist of laser odometry working at 5 Hz. Odometry constraints are calculated using simple scan-matching. The loop closing constraints comes for a Bag-of-words (BoW) system [16]. For the New College dataset, the odometry comes from stereo visual odometry and loop closures from the same BoW system. Loop closure verification for each dataset has been carried out using RRR [17], ensuring that all the loops present in the datasets are correct.

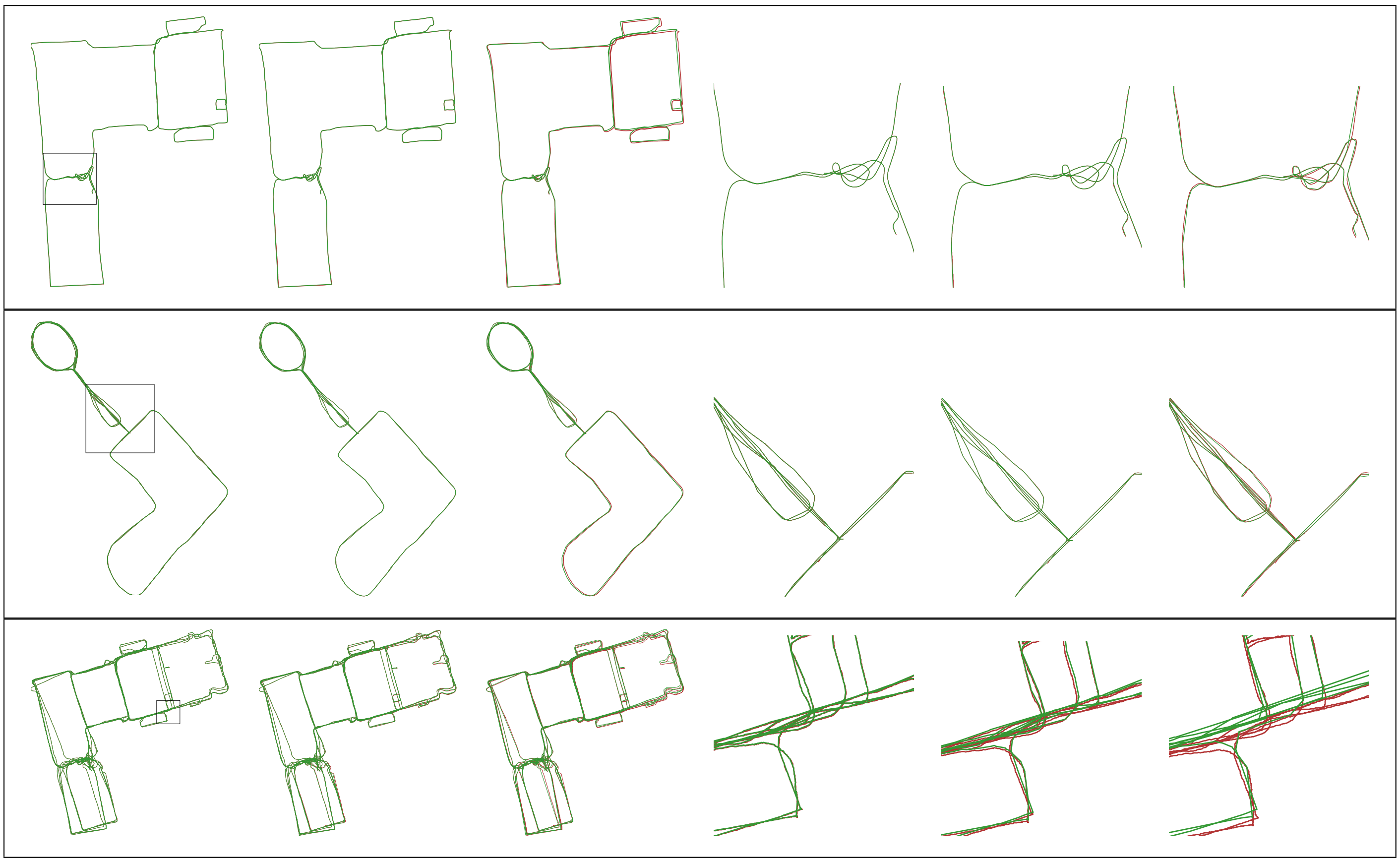

Fig. 2. Left to right: Maps built with the threshold set to 5cm, 10cm and 50cm respectively. Area marked in the first figure is shown zoomed in for each corresponding figure. Optimized complete graph is shown in red and reduced in green. (Best viewed online in color and zoomable)

	Bicocca			New College			Bicocca-multisession		
Threshold	Vertices	Time	Error(mean/med/max)	Vertices	Time	Error(mean/med/max)	Vertices	Time	Error(mean/med/max)
None	8358	0.429	-	10726	0.607	-	43116	2.3507	-
0.05m	1127	0.055	0.022/0.029/0.140	2389	0.141	0.013/0.005/1.716	8762	0.495	0.038/0.037/0.122
0.10m	864	0.049	0.040/0.051/0.227	1969	0.107	0.007/0.005/1.714	5308	0.299	0.124/0.100/0.640
0.50m	509	0.028	0.661/0.750/1.350	1357	0.080	0.332/0.087/1.467	2712	0.165	0.487/0.423/1.646

TABLE I

RESULTS FOR DIFFERENT THRESHOLD VALUES: TIME IN SECONDS AND ERROR IN METERS.

Effect on Error and timing

In order to investigate the effect of a given threshold on the error in optimized pose positions, we run the experiments with the mentioned datasets using three different thresholds, $0.05m$, $0.10m$ and $0.50m$. For all the cases, the time reported is for 10 Gauss-Newton iterations as given by g2o [4]. Error is calculated as the euclidean distance between the position of the node in the full-graph against the same node in the reduced graph. The results are given in Table I.

It can be seen that in every case, even a small threshold leads to a great reduction in the number of nodes needed to represent the graph. In case of Bicocca, we get an 86% reduction in the number of vertices and almost ten times speed up. Similarly a 77% and 79% reduction in the new college and Bicocca Multi-session dataset respectively.

The final estimated poses of the reduced graph and the full graph are very close to each other. For a trajectory length of 774m for Bicocca, the median error is about 3cm, and for New college with a trajectory length of 2.2km, the median error is about 0.5cm accompanied by a great reduction in the number of nodes.

As expected, increasing the threshold leads to fewer vertices remaining in the graph, but at the same time more error is introduced.

Fig. 2 shows the corresponding optimized reduced graphs. The difference between the full-graph (red) and reduced graph (red) is difficult to see in the original plots but is visible in the accompanying zoomed sections. Moving from left to right, more of the reduced graph becomes visible, meaning that it is starting to deviate from the estimate that is given

	Ratio	Count	
Threshold	min/mean/median/max	≤ 1	> 1
0.05m	0.96/1.00/1.00/1.07	642	483
0.10m	0.70/1.00/1.00/3.09	343	519
0.50m	0.52/0.99/0.99/2.10	343	164

TABLE II

EFFECT OF THRESHOLD: ESTIMATED UNCERTAINTY FOR BICOCCA DATASET.

by the full-graph.

Effect on Uncertainty estimates

We want to investigate how reducing the graph with different thresholds affects the estimates of marginal covariances of the nodes in the full-graph as compared to the nodes in the reduced-graph. We characterise the uncertainty of each marginal covariance matrix by its determinant. The determinant captures the overall uncertainty and is proportional to the total volume of the uncertainty ellipsoid. We compare the ratio of the determinant of the marginal covariance matrices of the corresponding nodes in the full-graph and reduced graph at different thresholds. The results for Bicocca dataset are show in Table II.

If the value of the ratio is less than one, it means that the reduced-graph over-estimates the uncertainty. On the other hand, a value greater than one suggests that the estimate might be over-confident. We prefer an estimate that is not over-confident. An under-confident (having more uncertainty than actual) is a more plausible, though less precise, solution.

In the results shown in Table. II, on average the uncertainty estimates are around or below 1. Even though the estimate for some nodes are over-confident, the amount by which they are away from the true value is small. This means that while reducing the nodes in this fashion leads to some estimates possibly becoming over-confident, the map is still usable. The distribution of these ratios can be seen in Fig. 3, which shows the majority of values tightly packed near 1 and very few above 1.

V. DISCUSSION

In this paper, we have presented a method for reducing pose graphs based on approximating the underlying trajectory using lines. Such approximation are used in a larger context by utilizing some property of the sensor such as range to ensure a minimum overlap or by using the distance travelled by the robot, all of which are done at the front-end of a SLAM system. The back-end does not need all the information to calculate a correct map estimate. With non-linear solver becoming more efficient, we are able to solve increasingly larger problems quickly, but the question still needs to be asked: How much information needs to go into the graph to get a good estimate of all the poses?

This question is important in the context of resource limited devices such as cell phones or development boards such as the raspberry pi. Such platforms have limited computation power and limited memory. Enabling such devices to perform on par with their high-end counterparts requires looking deeper into the problem and finding approximations that can work reasonably well on them.

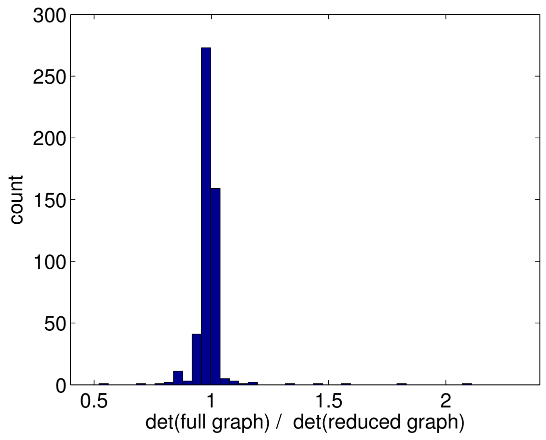

Fig. 3. Effect of reduction on the uncertainty estimates: Distribution of the ratio of determinants of marginal covariance for corresponding nodes in full graph and reduced-graph for Bicocca dataset at a threshold of 0.05 meters.

In this work we have shown a line based approximation method that drastically reduces the number of poses in the pose graph, enabling an order of a magnitude speed up with very little difference in terms of accuracy compared to the original problem. The method presented here works on the SLAM back-end and is therefore sensor agnostic. Similarly, techniques that do not require every pose in the graph, such as methods that reason about the validity of loop-closures [17]–[19], can greatly benefit from the presented approach.

As mentioned earlier, our method calculates a reduced version of the original pose graph, which is then optimized. We keep the original graph consisting of all the vertices and edges in the memory. This allows us to do an "on-demand" retrieval of the full-graph when it is needed for tasks such as navigation or reconstruction of the environment based on sensor data.

Given that all constraints in the graph are relative, once we optimize the reduced graph, constraints from the original-graph can be reintroduced and a full map estimate can be calculated. While we have shown this work as it applies to the problem of the optimization, it can be used as such to solve the problem of finding a good initial estimate for the complete graph. The advantage of using the reduced graph to calculate the optimal state arises from the fact that initially the graph may be far from the optimal state and suffer from slow convergence, which is made worse by the huge size of the problem. Solving a reduced problem leads to faster converge, as well as a better linearisation point when all (or a part of) the constraints are reintroduced. This leads to convergence in a very small number of iterations.

The method as presented here is suitable for non-holonomic robots, e.g. robots that can not rotate in-place. For holonomic robots, further constraints have to been introduced during the line estimation process for the special case of in-place rotation.

While we are representing the original problem with a reduced, approximated version, the number of nodes in the graph still grows with time. It is not desirable for long-term operation where the robot is expected to move around in the environment for a potentially infinite duration. Future work would include extending the current approach to take into account revisit information that is provided by loop-closures and finding a method that can fuse revisited segments of the map, giving a pose graph that grows with the area explored as opposed to growing with exploration time.

References

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

[2] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering," in *IEEE Intl. Conf. on Robotics and Automation, ICRA*, Shanghai, China, May 2011.

[3] G. Grisetti, R. Kümmerle, C. Stachniss, U. Frese, and C. Hertzberg, "Hierarchical optimization on manifolds for online 2d and 3d mapping," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, may 2010, pp. 273 –278.

[4] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, Shanghai, China, May 2011.

[5] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, ser. ISMAR '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 1–10. [Online]. Available: http://dx.doi.org/10.1109/ISMAR.2007.4538852

[6] ——, "Improving the agility of keyframe-based SLAM," in *Proc. 10th European Conference on Computer Vision (ECCV'08)*, Marseille, October 2008, pp. 802–815.

[7] V. Ila, J. M. Porta, and J. Andrade-Cetto, "Information-based compact pose slam," *Robotics, IEEE Transactions on*, vol. 26, no. 1, pp. 78–93, 2010.

[8] H. Kretzschmar, C. Stachniss, and G. Grisetti, "Efficient information-theoretic graph pruning for graph-based slam with laser range finders," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, sept. 2011, pp. 865 –871.

[9] H. Johannsson, M. Kaess, M. Fallon, and J. Leonard, "Temporally scalable visual SLAM using a reduced pose graph," in *RSS Workshop on Long-term Operation of Autonomous Robotic Systems in Changing Environments*, Sydney, Australia, Jul 2012.

[10] A. Walcott-Bryant, M. Kaess, H. Johannsson, and J. J. Leonard, "Dynamic Pose Graph SLAM: Long-term Mapping in Low Dynamic Environments," in *Proc. IEEE/RJS Int. Conference on Intelligent Robots and Systems*, 2012.

[11] H. Kretzschmar and C. Stachniss, "Information-theoretic compression of pose graphs for laser-based slam," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1219–1230, 2012.

[12] N. Carlevaris-Bianco and R. M. Eustice, " Generic factor-based node marginalization and edge sparsification for pose-graph SLAM ," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, May 2013.

[13] V. Nguyen, A. Martinelli, N. Tomatis, and R. Siegwart, "A comparison of line extraction algorithms using 2D laser rangefinder for indoor mobile robotics," in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*. IEEE, 2005, pp. 1929–1934.

[14] RAWSEEDS, "Robotics advancement through Webpublishing of sensorial and elaborated extensive data sets (project FP6-IST-045144)," 2009, http://www.rawseeds.org/rs/datasets.

[15] M. Smith, I. Baldwin, W. Churchill, R. Paul, and P. Newman, "The new college vision and laser data set," *The International Journal of Robotics Research*, vol. 28, no. 5, pp. 595–599, May 2009. [Online]. Available: http://www.robots.ox.ac.uk/NewCollegeData/

[16] C. Cadena, D. Gálvez-López, J. Tardós, and J. Neira, "Robust place recognition with stereo sequences," *IEEE Transaction on RObotics*, vol. 28, no. 4, pp. 871 –885, 2012.

[17] Y. Latif, C. Cadena, and J. Neira, "Robust Loop Closing Over Time," in *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2012.

[18] N. Sünderhauf and P. Protzel, "Switchable Constraints for Robust Pose Graph SLAM," in *Proc. IEEE/RJS Int. Conference on Intelligent Robots and Systems*, Vilamoura, Portugal, 2012.

[19] E. Olson and P. Agarwal, "Inference on networks of mixtures for robust robot mapping," in *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2012.