**2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan**

Long-Term Simultaneous Localization and Mapping with Generic Linear Constraint Node Removal

Nicholas Carlevaris-Bianco and Ryan M. Eustice

***Abstract*— This paper reports on the use of generic linear constraint (GLC) node removal as a method to control the computational complexity of long-term simultaneous localization and mapping. We experimentally demonstrate that GLC provides a principled and flexible tool enabling a wide variety of complexity management schemes. Specifically, we consider two main classes: batch multi-session node removal, in which nodes are removed in a batch operation between mapping sessions, and online node removal, in which nodes are removed as the robot operates. Results are shown for 34.9 h of real-world indoor-outdoor data covering 147.4 km collected over 27 mapping sessions spanning a period of 15 months.**

I. Introduction

Graph-based simultaneous localization and mapping (SLAM) [1]–[7] has been used to successfully solve many challenging SLAM problems in robotics. In graph SLAM, the problem of finding the optimal configuration of historic robot poses (and optionally the location of landmarks), is associated with a Markov random field or factor graph. In the factor graph representation, robot poses are represented by nodes and measurements between nodes by factors. Under the assumption of Gaussian measurement noise the graph represents a least squares optimization problem. The computational complexity of this problem is dictated by the density of connectivity within the graph, and by the number of nodes and factors it contains.

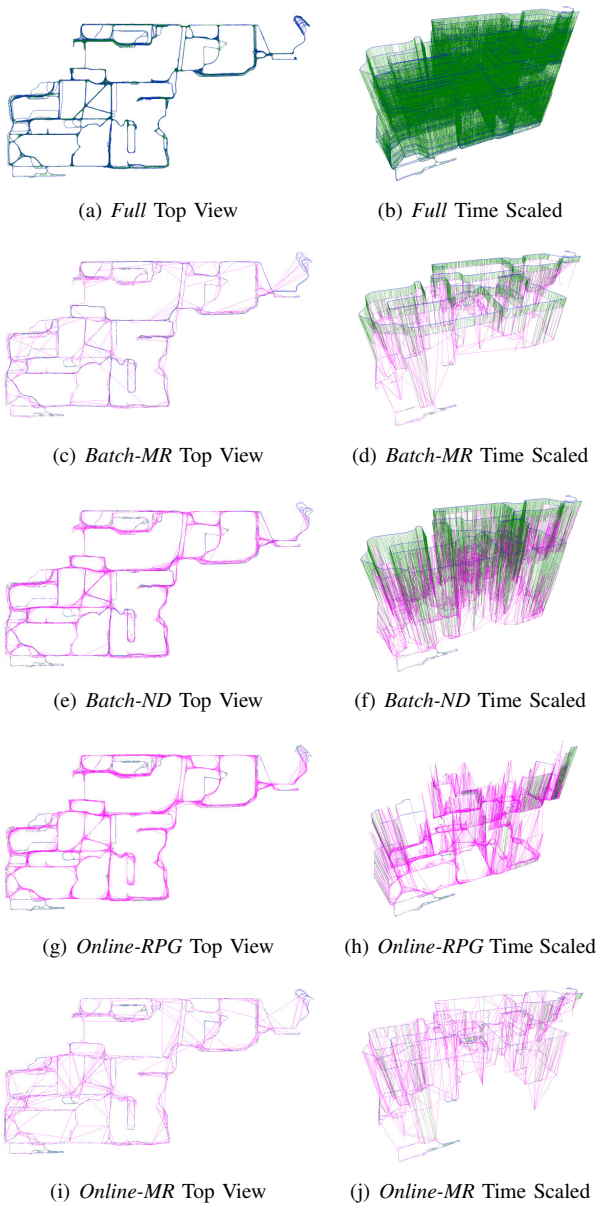
Unfortunately, the standard formulation of graph SLAM requires that nodes be continually added to the graph for localization. This is a problem for long-term applications as the computational complexity of the graph becomes dependent not only on the spatial extent of the environment, but also the *duration* of the exploration (Fig. 1(b)).

Early filtering-based works [8], [9], and more recently [10], have focused on controlling the computational complexity by enforcing sparse connectivity in the graph. In [11], an information-theoretic approach is used to slow the rate of the graph growth by avoiding the addition of uninformative poses. In [12], when the robot revisits a previously explored location, it avoids adding new nodes and instead adds links between existing nodes.

This work was supported in part by the National Science Foundation under award IIS-0746455, and in part by the Naval Sea Systems Command (NAVSEA) through the Naval Engineering Education Center under award N65540-10-C-0003.

N. Carlevaris-Bianco is with the Department of Electrical Engineering & Computer Science, University of Michigan, Ann Arbor, MI 48109, USA carlevar@umich.edu

R. Eustice is with the Department of Naval Architecture & Marine Engineering, University of Michigan, Ann Arbor, MI 48109, USA eustice@umich.edu



(a) *Full* Top View (b) *Full* Time Scaled

(c) *Batch-MR* Top View (d) *Batch-MR* Time Scaled

(e) *Batch-ND* Top View (f) *Batch-ND* Time Scaled

(g) *Online-RPG* Top View (h) *Online-RPG* Time Scaled

(i) *Online-MR* Top View (j) *Online-MR* Time Scaled

Fig. 1: The resulting graphs for 27 mapping sessions conducted over a period of 15 months using the proposed complexity management schemes (see Table I). Links include odometry (blue), 3D LIDAR scan matching (green) and generic linear constraints (magenta). The full graph without node removal is shown as *Full*. The left column shows a top down view. The right column shows an oblique view scaled by time in the z-axis; each layer along the z-axis represents a mapping session.

Recently, many works have proposed removing nodes from the SLAM graph as a means to control the computational complexity of the associated optimization problem [13]–[16]. In [13], the environment is spatially divided into neighborhoods and then a least-recently-used criteria is used to remove nodes with the goal of keeping a small set of example views that capture the changing appearance of the environment. In [15], nodes that provide the least information to an occupancy grid are removed. Nodes without associated imagery are removed in [14]. Finally, in [16], "inactive" nodes that no longer contribute to the laser-based map (because the environment has changed) are removed.

Each of the methods described in [12]–[16] provides insight into the question of which nodes should be removed from the graph. However, they all rely on pairwise measurement composition, as described in [17], to produce a new set of factors over the elimination clique (i.e., the nodes originally connected to the node being removed) after a node is removed from the graph.

Unfortunately, as shown in [18], pairwise measurement composition has two key drawbacks when used for node removal. First, it is not uncommon for a graph to be composed of many different types of "low-rank" constraints, such as bearing-only, range-only and other partial-state constraints. In these heterogeneous cases, measurement composition, if even possible, quickly becomes complicated as the constraint composition rules for all possible pairs of measurement types must be well defined. Second, the new constraints created by measurement composition are generally not independent (i.e., measurements may be double counted). This is acknowledged in [12] where an odometry link is discarded and the robot re-localized (along the lines of [9]) to avoid double counting measurements. Similarly, [16] uses a maximum of two newly composed constraints at the beginning and end of a "removal chain" (a sequence of nodes to remove) to ensure connectivity without double counting measurements. However, in general, double counting measurements may be unavoidable. As an example, consider removing node $\mathbf{x}_1$ from the graph in Fig. 2(a) using pairwise measurement composition while conforming to the sparsity pattern of the Chow-Liu tree (CLT) approximation, Fig. 2(c). Measurement composition will produce new measurements, $\mathbf{z}_{02} = \mathbf{z}_{01} \oplus \mathbf{z}_{12}$ and $\mathbf{z}_{03} = \mathbf{z}_{01} \oplus \mathbf{z}_{13}$, which double count $\mathbf{z}_{01}$ and are clearly not independent.

Methods that remove nodes without measurement composition have been proposed in [18]–[20]. These methods are based on replacing the factors in the marginalization clique with a linearized potential or a set of linearized potentials. In [19], these linearized potentials are refereed to as "star nodes." The dense formulation of our proposed generic linear constraint (GLC) [18] is essentially equivalent to "star nodes" while the sparse approximate GLC replaces the dense n-nary connectivity with an sparse tree structure. The method recently proposed in [20] again starts with a dense linear potential similar to star-nodes and dense-GLC but then approximates the potential through a $\mathcal{L}_1$-regularized guaranteed-consistent optimization to produce a sparse n-nary linear factor over the elimination clique.

Linearized potentials representing the result of marginalization are also used in [21] to reduce bandwidth while transmitting graphs between robots in a multi-robot distributed estimation framework. Nodes that are not part of the interaction between the robots' graphs are removed from linearized potentials, and a graph of these linearized potentials, referred to as a "summarized map", is transmitted between robots.

In this paper, we promote the use of GLC node removal [18] for long-term SLAM. GLC node removal shares many of the properties that make measurement composition appealing, while addressing heterogeneous graphs with non-full-state constraints and avoiding double counting measurement information. Our previous work, [18], demonstrated improvement in accuracy and consistency over pairwise measurement composition when performing large batch node removal operations. Here, we explore complexity management schemes that repeatedly apply GLC to remove nodes as the map is built. The core contributions of this paper are as follows:

- We provide an experimental evaluation of GLC node removal in both multi-session and online node removal schemes.
- We propose four complexity management schemes that can be implemented using GLC and validate them on a large long-term SLAM problem.
- We demonstrate a large long-term SLAM result with data collected over the course of 15 months and 27 mapping sessions.

The remainder of this paper is outlined as follows: In §II we review GLC node removal. We then propose several complexity management schemes that use GLC node removal in §III, which are experimentally evaluated in §IV. Finally, §V and §VI offer a discussion and concluding remarks.

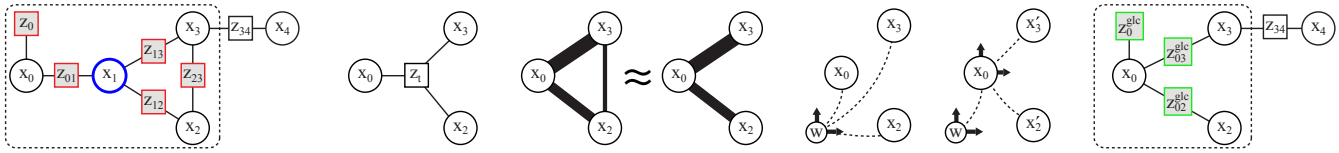
II. Generic Linear Constraint Node Removal

We first provide an overview of GLC node removal. For a full discussion and derivation we refer the reader to [18]. Here we focus on the CLT-based sparse-approximate version of GLC node removal as it is most appropriate for long term SLAM applications in which many more nodes are removed than kept. Using the dense-exact version of GLC node removal in these circumstances would produce graphs with a very high node degree, and therefore, a high computationally complexity because of the loss of sparsity.

Sparse-approximate GLC node removal is performed as follows:

1) The potential induced by marginalization over the elimination clique is computed.
2) The potential is approximated using a Chow-Liu tree.
3) The variables in the CLT potentials are reparameterized as relative transforms.
4) The CLT potentials are implemented as linear factors, referred to as generic linear constraints, replacing the original factors over the elimination clique in the graph.

An example of this process is shown in Fig. 2.



(a) Original Graph (b) Target Info. (c) Chow-Liu Tree (d) Root Shift (e) Final Graph

Fig. 2: Sample factor graph where node $\mathbf{x}_1$ is to be removed (a). Here $\mathbf{X}_m = [\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3]$. The factors $\mathbf{Z}_m = [\mathbf{z}_0, \mathbf{z}_{01}, \mathbf{z}_{12}, \mathbf{z}_{23}, \mathbf{z}_{13}]$ (highlighted in red in (a)) are those included in calculating the target information over the marginalization clique $\mathbf{X}_t = [\mathbf{x}_0, \mathbf{x}_2, \mathbf{x}_3]$ (b). The original distribution associated with the target information, $p(\mathbf{x}_0, \mathbf{x}_2, \mathbf{x}_3|\mathbf{Z}_m)$, is approximated using the Chow-Liu maximum mutual information spanning tree as $p(\mathbf{x}_0|\mathbf{Z}_m)p(\mathbf{x}_2|\mathbf{x}_0, \mathbf{Z}_m)p(\mathbf{x}_3|\mathbf{x}_0, \mathbf{Z}_m)$ (c). The pairwise potentials are reparameterized with respect to $\mathbf{x}_0$ to avoid linearization in the world-frame (d). New GLC factors are computed and inserted into the graph replacing $\mathbf{Z}_m$ (highlighted in green in (e)). Note that node removal only affects the nodes and factors withing the Markov blanket of $\mathbf{x}_1$ (dashed line).

A. Computing the Marginalization Potential

The first step in the algorithm is to correctly identify the potential induced in the graph by marginalization. This potential is characterized by its information matrix, which we refer to as the target information, Λ_t. Letting $\mathbf{X}_m \subset \mathbf{X}$ be the subset of nodes including the node to be removed and the nodes in its Markov blanket, and letting $\mathbf{Z}_m \subset \mathbf{Z}$ be the subset of measurement factors that only depend on the nodes in $\mathbf{X}_m$, we consider the distribution $p(\mathbf{X}_m|\mathbf{Z}_m) \sim \mathcal{N}^{-1}(\boldsymbol{\eta}_m, \Lambda_m)$. From Λ_m we can then compute the desired target information, Λ_t, by marginalizing out the elimination node using the standard Schur-complement form [5]. An example graph is shown in Fig. 2(a) and Fig. 2(b).

The key observation here is that, because marginalization only affects the elimination clique, a factor, or set of factors, which induces the potential characterized by Λ_t will induce the same potential as true node marginalization at the given linearization point.

It is important to note that the constraints in $\mathbf{Z}_m$ may be purely relative and/or low-rank (e.g., bearing or range-only) and, therefore, may not fully constrain $p(\mathbf{X}_m|\mathbf{Z}_m)$. As a result, Λ_t, and distributions derived from Λ_t, may be low rank. This will be accounted for by GLC in §II-D.

B. Chow-Liu Tree Approximation of the Marginalization Potential

Unfortunately, inducing the marginalization potential exactly requires an n-ary factor with dense connectivity amongst the nodes in the Markov blanket. This leads to fill-in that can greatly increase the computational complexity of optimizing the graph [2], [7]. In [15], Kretzschmar and Stachniss insightfully propose the use of a CLT [22] to approximate the individual elimination cliques as sparse tree structures. Using GLC we can implement the CLT approximation without the aforementioned pitfalls of measurement composition and provide a significant improvement in performance over [15], as shown in [18].

The CLT approximates a joint distribution as the product of pairwise conditional distributions,

$$p(\mathbf{x}_1, \cdots, \mathbf{x}_n) \approx p(\mathbf{x}_1) \prod_{i=2}^{n} p(\mathbf{x}_i|\mathbf{x}_{\mathrm{p}(i)}), \tag{1}$$

where $\mathbf{x}_1$ is the root variable of the CLT and $\mathbf{x}_{\mathrm{p}(i)}$ is the parent of $\mathbf{x}_i$. The pairwise conditional distributions are selected such that the Kullback-Leibler divergence (KLD) between the original distribution and the CLT approximation is minimized. To construct the CLT, the maximum spanning tree over all possible pairwise mutual information pairings is found (Fig. 2(c)). Because we wish to reproduce the marginalization potential we compute the CLT approximation (1) over Λ_t, and will represent the CLT's unary and binary potentials as new GLC factors.

We first consider CLT binary potentials, $p(\mathbf{x}_i|\mathbf{x}_{\mathrm{p}(i)})$, and in the following use $\mathbf{x}_j = \mathbf{x}_{\mathrm{p}(i)}$ as shorthand for the parent node of $\mathbf{x}_i$. Note that the joint marginal, $p_t(\mathbf{x}_i, \mathbf{x}_j)$, is computed directly from Λ_t[1]. From this joint marginal, we can then easily write the desired conditional, $p_t(\mathbf{x}_i|\mathbf{x}_j) = \mathcal{N}(\boldsymbol{\mu}_{i|j}, \Sigma_{i|j}) \equiv \mathcal{N}^{-1}(\boldsymbol{\eta}_{i|j}, \Lambda_{i|j})$, and express it as a constraint as

$$\mathbf{e} = \mathbf{x}_i - \boldsymbol{\mu}_{i|j} = \mathbf{x}_i - \Lambda_{ii}^{-1}(\boldsymbol{\eta}_i - \Lambda_{ij}\mathbf{x}_j), \tag{2}$$

where $\mathbf{e} \sim \mathcal{N}^{-1}(\mathbf{0}, \Lambda_{i|j})$, and with Jacobian,

$$\mathrm{E} = \begin{bmatrix} \frac{\partial \mathbf{e}}{\partial \mathbf{x}_i} & \frac{\partial \mathbf{e}}{\partial \mathbf{x}_j} \end{bmatrix} = \begin{bmatrix} \mathrm{I} & \Lambda_{ii}^{-1}\Lambda_{ij} \end{bmatrix}. \tag{3}$$

Therefore, using the standard information-form measurement update, we see that this constraint adds information

$$\mathrm{E}^\top \Lambda_{i|j} \mathrm{E}, \tag{4}$$

where $\Lambda_{i|j}$ is simply Λ_{ii}.

Similarly, the CLT's root unary potential, $p_t(\mathbf{x}_1) = \mathcal{N}^{-1}(\boldsymbol{\eta}_1, \Lambda_1)$, can be computed from Λ_t by marginalization.

C. Reparametrizing the CLT Potentials

We now have a set of unary and binary potentials that we wish to implement as factors in the graph. For each potential, let its support variables be denoted as $\mathbf{x}_i$ and its associated information as Λ_i.

Creating GLC factors directly from Λ_i over $\mathbf{x}_i$ will commit to linearization points in the world-frame for the nodes in $\mathbf{x}_i$. This may be acceptable in applications where good world-frame linearization points are known prior to marginalization; however, a more tenable assumption is that a good linearization point exists for the local *relative-frame* transforms between nodes within the elimination clique.

[1]In this section, we refer to marginal and conditional distributions with respect to the target information, Λ_t, *not* the distribution represented by the full graph.

To account for this in [18] we defined a "root-shift" function that reparameterizes the variables that support a general n-ary potential in relative-frame coordinates instead of in the world-frame. For the CLT approximation, we only root-shift the binary potentials, as it provides no benefit for unary potentials.

For a binary potential over nodes in the world-frame, $\mathbf{x}_i = [\mathbf{x}_1^w, \mathbf{x}_2^w]$, the root-shift function becomes

$$\mathbf{x}_i' = \begin{bmatrix} \mathbf{x}_w^a \\ \mathbf{x}_2^a \end{bmatrix} = r\left(\begin{bmatrix} \mathbf{x}_1^w \\ \mathbf{x}_2^w \end{bmatrix}\right) = \begin{bmatrix} \ominus \mathbf{x}_1^w \\ \ominus \mathbf{x}_1^w \oplus \mathbf{x}_2^w \end{bmatrix}. \tag{5}$$

In this function, the first node is arbitrarily chosen as the root. The inclusion of the inverse of the root pose, $\mathbf{x}_w^1$, is important as it ensures that the Jacobian of the root-shift operation, R, is invertible, and allows for the representation of target information that is not purely relative.

The root-shifted information, Λ_i', associated with the potential can be calculated as

$$\Lambda_i' = \mathrm{R}^{-\top} \Lambda_i \mathrm{R}^{-1}. \tag{6}$$

Now when a GLC factor is created, the linearized constraint will be embedded within a relative coordinate frame with respect to the first node in the potential, as opposed to an absolute world coordinate frame (Fig. 2(d)).

D. Generic Linear Constraints

We now have a set of unary and root-shifted binary potentials that we wish to implement as factors in the graph. For simplicity of notation, let each potential's support variables again be denoted as $\mathbf{x}_i$ and its associated information as Λ_i. If we consider an observation model that directly observes $\mathbf{x}_i$ with a measurement uncertainty that is defined by Λ_i,

$$\mathbf{z} = \mathbf{x}_i + \mathbf{w} \;\text{ where }\; \mathbf{w} \sim \mathcal{N}^{-1}\big(\mathbf{0}, \Lambda_i\big), \tag{7}$$

and set the measurement value, $\mathbf{z}$, equal to the current linearization point of the support variables, $\hat{\mathbf{x}}_i$, we will induce the desired potential in the graph.

Unfortunately, as mentioned in §II-A, because Λ_i is derived from the target information it may not be full-rank, which is problematic for optimization methods that rely upon a square root factorization of the measurement information matrix [2], [7]. We can, however, use principle component analysis to transform the measurement to a lower-dimensional representation that is full-rank.

Due to the nature of its construction, Λ_i will be a real, symmetric, positive semi-definite matrix with an eigen-decomposition given by

$$\Lambda_i = \begin{bmatrix} \mathbf{u}_1 & \cdots & \mathbf{u}_q \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \lambda_q \end{bmatrix} \begin{bmatrix} \mathbf{u}_1^\top \\ \vdots \\ \mathbf{u}_q^\top \end{bmatrix} = \mathrm{U}\mathrm{D}\mathrm{U}^\top, \tag{8}$$

where U is a $p \times q$ matrix, D is a $q \times q$ matrix, p is the dimension of Λ_i, and $q = \mathrm{rank}(\Lambda_i)$. Letting $\mathrm{G} = \mathrm{D}^{\frac{1}{2}}\mathrm{U}^\top$ allows us to write a transformed observation model,

$$\mathbf{z}_{glc} = \mathrm{G}\mathbf{z} = \mathrm{G}\mathbf{x}_i + \mathbf{w}' \;\text{ where }\; \mathbf{w}' \sim \mathcal{N}^{-1}\big(\mathbf{0}, \Lambda'\big). \tag{9}$$

Using the pseudo-inverse [23], $\Lambda_i^+ = \mathrm{U}\mathrm{D}^{-1}\mathrm{U}^\top$, and noting that $\mathrm{U}^\top\mathrm{U} = \mathrm{I}_{q\times q}$, we find that

$$\Lambda' = (\mathrm{G}\Lambda_i^+\mathrm{G}^\top)^{-1} = (\mathrm{D}^{\frac{1}{2}}\mathrm{U}^\top(\mathrm{U}\mathrm{D}^{-1}\mathrm{U}^\top)\mathrm{U}\mathrm{D}^{\frac{1}{2}})^{-1} = \mathrm{I}_{q\times q}.$$

This GLC factor will contribute the desired information back to the graph, i.e.,

$$\mathrm{G}^\top\Lambda'\mathrm{G} = \mathrm{G}^\top\mathrm{I}_{q\times q}\mathrm{G} = \Lambda_i,$$

but is itself non-singular. This is a key advantage of the proposed GLC method; it automatically determines the appropriate measurement rank such that Λ' is $q \times q$ and invertible, and G is a $q \times p$ new observation model that maps the p-dimensional state to the q-dimensional measurement.

The node can then be removed and the existing factors replaced with the new set of constraints spanning the elimination clique (Fig. 2(e)).

III. Graph Complexity Management through Node Removal

We now propose four GLC-based graph management schemes; two that are performed as a batch step between each mapping session, and two that remove nodes as the robot moves through the environment. In each case, we attempt to produce a graph that has a complexity dictated primarily by spatial extent and not by mapping duration. We therefore seek to remove spatially redundant nodes. The definition of spatially redundant nodes varies depending on the motion constraints of the robot and its sensing modalities. In our experiments, we used data from a ground robot with 3D LIDAR as the primary sensing modality. Therefore, we will only consider translation in the ground plane for adding new nodes—whenever odometry indicates that the robot has translated more than $3\ \mathrm{m}$. Similarly, we consider nodes within $3\ \mathrm{m}$ of each other to be spatially redundant.

In the following sections, we will refer to each of the experimental complexity schemes using the abbreviated names in Table I.

TABLE I: Experimental Complexity Management Schemes

Scheme	Description
Batch-MR	Batch - Keep most recent (Alg. 1)
Batch-ND	Batch - Keep highest node degree (Alg. 2)
Online-RPG	Online - Emulate reduced pose graph (Alg. 3)
Online-MR	Online - Keep most recent (Alg. 4)

A. Batch Multi-Session Node Removal

We first consider a multi-session scenario where the robot repeatedly performs SLAM in discrete sessions. Under these conditions node removal can be performed between sessions as a batch operation. In oversampled regions, we seek to keep the most recently added nodes as this allows the map to adapt to the changing environment. For example, during our data collection there were several locations undergoing construction. As we pass through these regions multiple times, old nodes that are spatially redundant should be removed, while new nodes capturing the changing structure should be kept. Essentially, the map can march forward in

Algorithm 1 Batch Node Removal: Keep most recent

Given nodes in graph, $nodes = \{n_0 \dots n_m\}$
$nodes = \texttt{sort\_by\_time\_descending}(nodes)$
$keep = \{n_0\}$
for all n_i in $nodes$ **do**
  if $\texttt{is\_not\_spatially\_redudant}(n_i, keep)$ **then**
    $keep = keep \cup n_i$
  end if
end for
$GLC\_remove(nodes \setminus keep)$

Algorithm 2 Batch Node Removal: Keep highest node degree

Given all m nodes in graph, $nodes = \{n_0 \dots n_m\}$
$nodes = \texttt{sort\_by\_node\_degree\_descending}(nodes)$
$keep = \{n_0\}$
for all n_i in $nodes$ **do**
  if $\texttt{is\_not\_spatially\_redudant}(n_i, keep)$ **then**
    $keep = keep \cup n_i$
  end if
end for
$GLC\_remove(nodes \setminus keep)$

time, replacing old observations with new ones. In practice this is performed by sorting the nodes according to their instantiation time, and then looping through the nodes in order, keeping each node that is "sufficiently far" from all nodes currently being kept, as detailed in Algorithm 1.

This strategy, however, only considers when a node was added. Depending on the application, additional information about each node may be available and may lead to different criteria. As an example, we consider that some nodes in the graph may be more useful for registration than others. Nodes that have been successfully registered against many times may be good candidates to keep in the graph; first, because they occur in locations that the robot repeatedly visits and second, because the observation is such that registration is repeatedly successful. Therefore, if we first sort by node degree (the number of edges connected to a node) and then remove nodes as before, we have a strategy that encourages the retention of useful nodes, as detailed in Algorithm 2. Note that many nodes will have the same node degree and in the case of a tie, instantiation time is used as a secondary criteria.

B. Online Node Removal

The first online node removal scheme we consider emulates the complexity reduction scheme proposed in [12]; but using GLC instead of measurement composition. In this method, referred to as the "Reduced Pose Graph", a new node is not added when the current pose is spatially redundant. Instead, the measurements from the current pose are used to add constraints between existing nodes. Conceptually, this can be thought of as temporarily adding the current node to the graph, adding its measurements and then marginalizing out the current node. This practice will add constraints between the existing nodes without permanently adding a new redundant node. We can, therefore, easily emulate this strategy by performing GLC node removal

Algorithm 3 Online Node Removal: Emulate reduced pose graph

Given previous nodes in graph, $nodes = \{n_0 \dots n_m\}$
Given recent node n_{m+1}
$neighbors = \texttt{get\_redundant\_neighbors}(n_{m+1}, nodes)$
if $neighbors \neq \varnothing$ **then**
  $GLC\_remove(n_{m+1})$
end if

Algorithm 4 Online Node Removal: Keep most recent

Given previous nodes in graph, $nodes = \{n_0 \dots n_m\}$
Given recent node n_{m+1}
$neighbors = \texttt{get\_redundant\_neighbors}(n_{m+1}, nodes)$
$GLC\_remove(neighbors)$

to remove recently-added redundant nodes, as detailed in Algorithm 3. Given a recently added node we look for its spatially redundant neighbors (i.e., nodes that are sufficiently close to the new node so as to make it redundant). If any spatially redundant neighbors are found, the recently-added node is removed.

From a data association perspective, the Reduced Pose Graph formulation may not be ideal as it keeps the first sensory sample of a given location and avoids adding all subsequent observations. We therefore consider an online scheme that does exactly the opposite; instead of removing the recent node, we remove the neighbors that are made redundant by the recent node, as detailed in Algorithm 4. This is essentially an online version of Algorithm 1.

IV. Experimental Results

In order to validate the proposed GLC complexity management schemes, we performed experiments using data collected over the course of 15 months using a Segway robotic platform (Fig. 3). Data was collected over 27 trials (approximately bi-weekly) between January 8$^{\text{th}}$, 2012 and April 5$^{\text{th}}$, 2013 by manually driving the robot through the University of Michigan's North Campus. Data was collected both indoors and outdoors, at varying times of the day and in

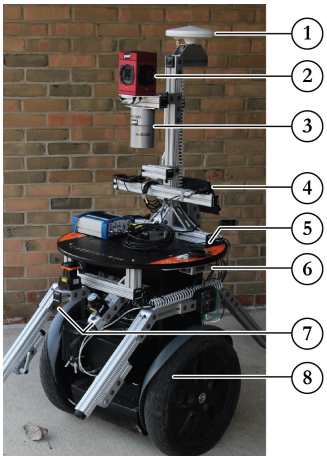


Fig. 3: Segway robotic platform used for experimental data collection. Outfitted with an RTK GPS (for ground-truth) (1), omnidirectional camera (2), 3D LIDAR (3), Attitude sensor (4), consumer-grade GPS (5), 1-axis FOG (6), 2D LIDARs (7), and CPU (8).

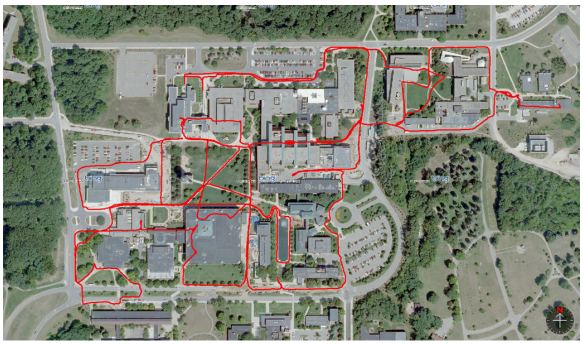


Fig. 4: Sample trajectory from one session of data collection, overlaid on satellite imagery.

the presence of dynamic elements including people and cars. Additionally, the data set contains several large construction projects. Each trial through the environment is on average 1.29 h in duration and 5.5 km in length totaling 34.9 h of data covering 147.4 km (Fig. 4).

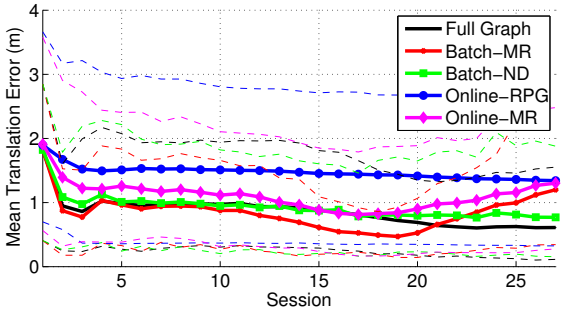
In all experiments, graph constraints are derived from odometry, 3D LIDAR scan matching [24] and, when available, consumer-grade GPS. GLC was implemented using iSAM [7], [25] as the underlying optimization engine. The code is available for download within the iSAM repository [26].

A ground-truth graph was created from all trajectories without node removal and with the addition of constraints from a highly accurate RTK GPS system.

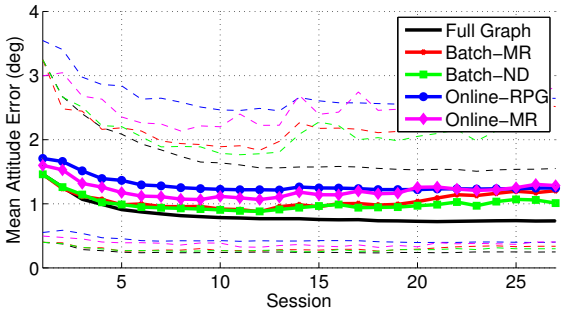
The graphs for each proposed method, at the end of the last full run, are shown in Fig. 1. In the right column, by scaling the z-axis according to time, we can clearly see the effects of the different node removal schemes. Using *Batch-MR* we see that the most recent session is well connected to the previous session with some sparse connectivity to older nodes in the graph. *Batch-ND* produces similar results but with more connectivity to previous nodes, which have been kept due to a high node degree. *Online-MR* has also removed the bulk of the nodes from previous sessions, additionally removing those from the penultimate session. In contrast, *Online-RPG* has kept its earliest observations of each location and removed newer nodes adding connectivity between older nodes.

A. Error with respect to Ground-Truth

First, we consider the performance of each complexity management scheme in terms of translation and attitude error from the ground-truth. We include a full graph that was built without node removal as a baseline (Fig. 5). We see all methods produce estimates with error similar to the full graph, with the online methods having slightly higher error than the batch methods in general. It is worth noting that by the end of each session, before batch sparsification, the batch methods will have almost double the number of nodes as the online methods, potentially allowing for more informative loop closures (Fig. 6(a)). The *Online-RPG* method produces the highest error. This is most likely due to the fact that data association becomes more difficult as the environment changes with time, as described in §III-B.



(a) Translation Error



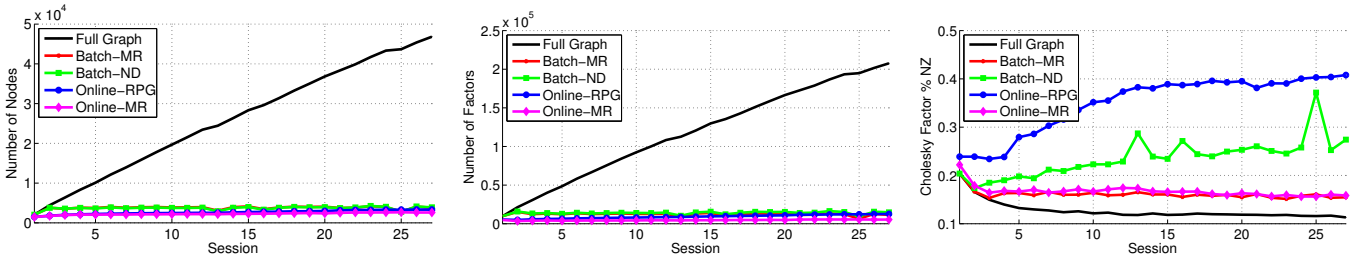
(b) Attitude Error

Fig. 5: Mean error for translation ($\sqrt{\delta_x^2 + \delta_y^2 + \delta_z^2}$) and attitude ($\sqrt{\delta_r^2 + \delta_p^2 + \delta_h^2}$) with respect to RTK-based ground-truth at the end of each mapping session for batch and online node removal methods. 5% and 95% percentile bounds are denoted with dashed lines. Errors for the node removal methods are compared against the errors for the full graph from which no nodes were removed (black).

B. Computational Complexity

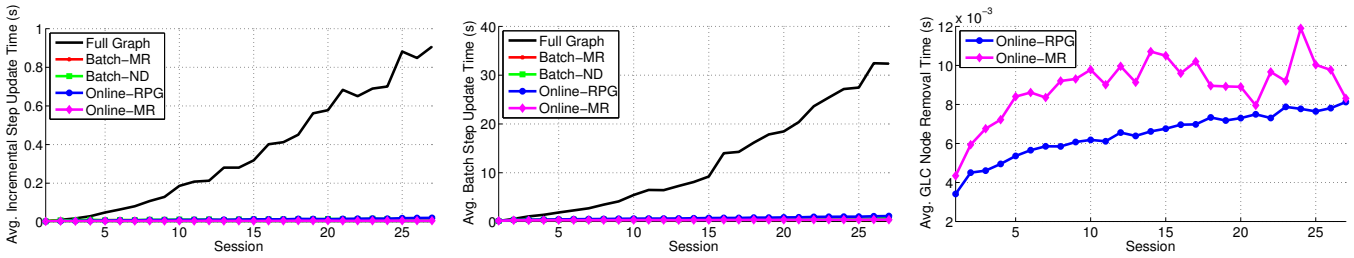
Though the graphs produced with GLC node removal have a similar or slightly higher error than the full graph, they are vastly less computationally complex. We see in Fig. 6 that all schemes limit the number of nodes and factors to be essentially constant as only small additions to the spatial extent of the map are made after the first session, with no method exceeding 4,000 nodes or 15,000 factors. In comparison, the full graph grows linearly ending with over 46,000 nodes and 200,000 factors. We also see that the sparsity of the measurement Cholesky Factor, R, is nearly constant, with *Online-RPG* growing the fastest as new connectivity is added between old nodes when newer nodes are removed. Note, however, that even for *Online-RPG* the maximum fill in is 0.4%, still quite sparse.

As new nodes and factors are added to the graph, iSAM performs two different types of updates; an incremental update, where the solution is updated without relinearization, and a batch update, where the solution is repeatedly relinearized and solved until convergence. In our experiments the batch optimization update was called every 50 incremental updates. In Fig. 7, we see that in the full graph, the computation time for incremental and batch update steps grows super-linearly, while for the proposed methods they remain roughly constant (Fig. 7(a) and 7(b)). The time to remove a node using GLC is also relatively constant and on the order of 10 ms (Fig. 7(c)), though slightly higher in the case of



(a) Number of Nodes (b) Number of Factors (c) Graph Sparsity

Fig. 6: Graph complexity. Note that for batch methods the complexity statistics are recorded at the end of each session immediately before node removal.



(a) Incremental Updates (b) Batch Updates (c) GLC Node Removal

Fig. 7: Mean CPU time for incremental and batch iSAM optimization update steps, and for GLC node removal, in seconds.

Online-RPG due to the higher connectivity density.

It is important to note that not all methods perform the same number of incremental and batch update steps. iSAM requires a batch optimization step after node removal and it is desirable to be as close to the optimal as possible before creating new GLC constraints. Therefore, in the *Online-RPG* and *Online-MR* schemes, we wait until 10 nodes have been flagged for removal, and then perform a batch-relinearization optimization step immediately before and after removing them. This results in the batch optimization step being called more often for the *Online-RPG* and *Online-MR* schemes. The total processing time for the 34.9 h of logged data, including graph optimization, node removal, data association and scan matching, took 58.7 h for the full graph. When using the proposed complexity management schemes total processing times were reduced to 6.1 h for *Batch-MR*, 6.3 h for *Batch-ND*, 7.9 h for *Online-RPG*, and 6.8 h for *Online-MR*, which is at least 4.4 times faster than real-time.

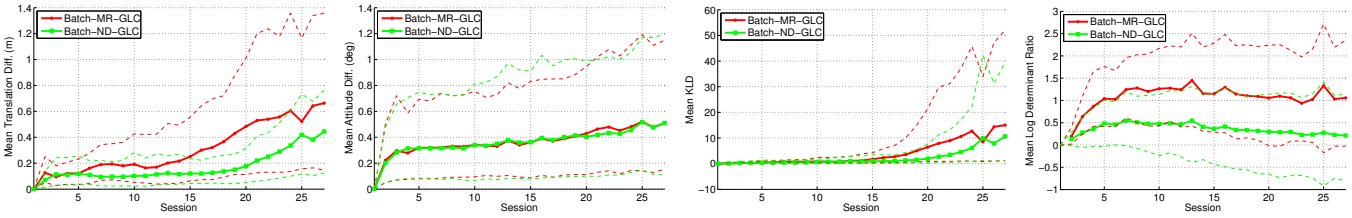
C. Distribution Comparison

In the previous experiment, each complexity management scheme elects to remove a different set of nodes and, therefore, the robot will make different data association decisions, resulting in fundamentally different graphs. In order to isolate the effects of GLC, we wish to directly compare the distribution produced by repeatedly applying sparse-approximate GLC node removal to a full distribution derived using the exact same measurements, from which no nodes have been removed. This can be done for the batch methods by accumulating the measurements from each session into one large graph. The results of this comparison are shown in Fig. 8. Here we see that repeatedly applying sparse approximate GLC node removal will produce a difference in the estimates from the full graph, though the difference remains low, both in terms of mean (Fig. 8(a) and 8(b)) and KLD (Fig. 8(c)). By looking at the log-ratio of the marginal covariance determinants estimated in the reduced and full graph, we can see that the reduced graph is neither systematically conservative (strongly positive values) nor systematically overconfident (strongly negative values) (Fig. 8(d)). This is due to the fact that the CLT approximation simply seeks to produce the minimum KLD and does not guarantee a conservative approximation. Guaranteeing a conservative approximation, while still producing a low KLD, remains as future work for GLC.

V. Complexity Management Scheme Design Considerations

Having compared four different complexity management schemes based on GLC node removal we can highlight some things to consider when designing new schemes:

- Removing larger sets of nodes less often produces better results than removing small sets of nodes more often. Note that there is not really a binary difference between online and batch node removal, it is just a matter of how long nodes are left in the graph before removal.
- Even though the GLC constraints are reparameterized in terms of relative transforms they still commit to a relative linearization point. Therefore, it is desirable that the relative transforms be as close as possible to the optimal solution before node removal. The graph should be optimized as well as possible before node removal.
- When removing a set of nodes it is important to note that the order in which they are removed effects the resulting graph connectivity. Experimentally, we found that removing long chains of nodes sequentially



(a) Translation Error (b) Attitude Error (c) KL Divergence (d) $\log(|\Sigma_{ii}^{GLC}|/|\Sigma_{ii}^{Full}|)$

Fig. 8: Comparison of the estimated distributions using batch methods with the estimated distributions using the same measurements but without node removal. Translation error (a) is defined as $\sqrt{\delta_x^2 + \delta_y^2 + \delta_z^2}$ and attitude error (b) as $\sqrt{\delta_r^2 + \delta_p^2 + \delta_h^2}$. The average KLD between resulting marginal covariances for each node are shown in (c). By looking at the log-ratio of the marginal covariances determinants (d) we can see that the reduced graph is neither systematically conservative (strongly positive values) nor systematically overconfident (strongly negative values). 5% and 95% percentile bounds are denoted with dashed lines.

sometimes produced large star shaped trees in the graph. To avoid this, sets of nodes were removed in a randomized order in all experiments. The variable elimination ordering problem [27] is well studied for dense node removal. The application and adaptation of existing variable elimination ordering strategies for node removal with sparse connectivity could further improve the performance of GLC-based complexity management schemes.

VI. Conclusions

In this paper, we demonstrated that GLC provides a principled and flexible tool enabling a variety of complexity management schemes where pairwise measurement composition would normally be used. We proposed and evaluated four complexity management schemes based on GLC node removal. Each method is shown to successfully solve a large-scale long-term SLAM while greatly reducing the associated computational complexity.

References

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

[2] F. Dellaert and M. Kaess, "Square root SAM: Simultaneous localization and mapping via square root information smoothing," *Int. J. Robot. Res.*, vol. 25, no. 12, pp. 1181–1203, 2006.

[3] S. Thrun and M. Montemerlo, "The graph SLAM algorithm with applications to large-scale mapping of urban structures," *Int. J. Robot. Res.*, vol. 25, no. 5-6, pp. 403–429, 2006.

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

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

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

[7] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Trans. Robot.*, vol. 24, no. 6, pp. 1365–1378, Dec. 2008.

[8] S. Thrun, Y. Liu, D. Koller, A. Ng, Z. Ghahramani, and H. Durrant-Whyte, "Simultaneous localization and mapping with sparse extended information filters," *Int. J. Robot. Res.*, vol. 23, no. 7-8, pp. 693–716, July-Aug. 2004.

[9] M. R. Walter, R. M. Eustice, and J. J. Leonard, "Exactly sparse extended information filters for feature-based SLAM," *Int. J. Robot. Res.*, vol. 26, no. 4, pp. 335–359, Apr. 2007.

[10] J. Vial, H. Durrant-Whyte, and T. Bailey, "Conservative sparsification for efficient and consistent approximate estimation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots and Syst.*, 2011, pp. 886–893.

[11] V. Ila, J. M. Porta, and J. Andrade-Cetto, "Information-based compact pose SLAM," *IEEE Trans. Robot.*, vol. 26, no. 1, pp. 78–93, Feb. 2010.

[12] H. Johannsson, M. Kaess, M. Fallon, and J. J. Leonard, "Temporally scalable visual SLAM using a reduced pose graph," in *Proc. IEEE Int. Conf. Robot. and Automation*, May 2013, pp. 54–61.

[13] K. Konolige and J. Bowman, "Towards lifelong visual maps," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots and Syst.*, 2009, pp. 1156–1163.

[14] E. Eade, P. Fong, and M. E. Munich, "Monocular graph SLAM with complexity reduction," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots and Syst.*, 2010, pp. 3017–3024.

[15] H. Kretzschmar and C. Stachniss, "Information-theoretic compression of pose graphs for laser-based SLAM," *Int. J. Robot. Res.*, vol. 31, pp. 1219–1230, 2012.

[16] 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/RSJ Int. Conf. Intell. Robots and Syst.*, 2012, pp. 1871–1878.

[17] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," in *Autonomous Robot Vehicles*, I. Cox and G. Wilfong, Eds. Springer-Verlag, 1990, pp. 167–193.

[18] N. Carlevaris-Bianco and R. M. Eustice, "Generic factor-based node marginalization and edge sparsification for pose-graph SLAM," in *Proc. IEEE Int. Conf. Robot. and Automation*, May 2013, pp. 728–735.

[19] J. Folkesson and H. Christensen, "Graphical SLAM—A self-correcting map," in *Proc. IEEE Int. Conf. Robot. and Automation*, New Orleans, LA, April 2004, pp. 383–390.

[20] G. Huang, M. Kaess, and J. J. Leonard, "Consistent sparsification for graph optimization," in *Proc. European Conf. Mobile Robotics*, Barcelona Spain, September 2013.

[21] A. Cunningham, V. Indelman, and F. Dellaert, "DDF-SAM 2.0: Consistent distributed smoothing and mapping," in *Proc. IEEE Int. Conf. Robot. and Automation*, Karlsruhe, Germany, May 2013.

[22] C. Chow and C. N. Liu, "Approximating discrete probability distributions with dependence trees," *IEEE Trans. on Info. Theory*, vol. 14, pp. 462–467, 1968.

[23] C. R. Rao and S. K. Mitra, *Generalized Inverse of Matrices and its Applications*. John Wiley & Sons, 1971.

[24] M. Magnusson, "The three-dimensional normal-distributions transform – an efficient representation for registration, surface analysis, and loop detection," Ph.D. dissertation, Örebro University, 2009.

[25] M. Kaess and F. Dellaert, "Covariance recovery from a square root information matrix for data association," *Robot. and Autonmous Syst.*, vol. 57, pp. 1198–1210, Dec. 2009.

[26] M. Kaess, H. Johannsson, D. Rosen, N. Carlevaris-Bianco, and J. Leonard, "Open source implementation of iSAM," http://people.csail.mit.edu/kaess/isam, 2010.

[27] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.