Temporally Scalable Visual SLAM using a Reduced Pose Graph

Hordur Johannsson, Michael Kaess, Maurice Fallon and John J. Leonard

***Abstract*— In this paper, we demonstrate a system for temporally scalable visual SLAM using a reduced pose graph representation. Unlike previous visual SLAM approaches that maintain static keyframes, our approach uses new measurements to continually improve the map, yet achieves efficiency by avoiding adding redundant frames and not using marginalization to reduce the graph. To evaluate our approach, we present results using an online binocular visual SLAM system that uses place recognition for both robustness and multi-session operation. Additionally, to enable large-scale indoor mapping, our system automatically detects elevator rides based on accelerometer data. We demonstrate long-term mapping in a large multi-floor building, using approximately nine hours of data collected over the course of six months. Our results illustrate the capability of our visual SLAM system to map a large are over extended period of time.**

I. Introduction

To achieve long-term robotic autonomy, in complex and dynamic environments, mapping algorithms are required that scale solely with the area explored and are independent of the duration of exploration and operation. There are many applications for autonomously navigating mobile robots, such as service, delivery, and search and rescue, in which long-term mapping and localization operations are critical. To be widely applicable, the system should construct the map of its operating environment using only onboard sensors, continuously updating and extending this map with new information.

Many recent solutions to mapping are based on the pose graph formulation [18]. In this formulation the world is represented by a set of discrete poses sampled along the full trajectory of the robot, which are connected by odometry and loop closure constraints. Very efficient recursive algorithms have been presented which can maintain an online solution to this continuously expanding optimization problem, however the pose graph, by design, grows unbounded in time [1]. This is true even for small environments which are repeatedly explored, making the naive application of the pose graph unsuitable for long-term mapping.

It is desirable to achieve a persistent mapping solution that scales only in terms of the spatial extent of an environment, and not the duration of the mission. Additionally, long-term persistent operation will require the ability to develop compact representations, which can effectively describe an environment of interest, yet still provide robustness to changes in the environment and recovery from mistakes.

This work was partially supported by ONR grants N00014-10-1-0936, N00014-11-1-0688, and N00014-12-10020. The authors are with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA `hordurj, kaess, mfallon, jleonard@mit.edu`

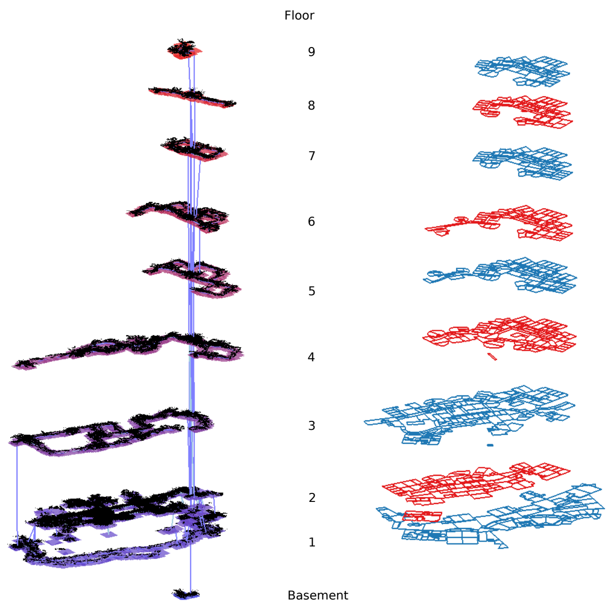


Fig. 1. Left: A map of ten floors of the MIT Stata Center created using the reduced pose graph in combination with a real-time visual SLAM system. The data used was collected in 14 sessions spanning a six month period. The total operation time was nine hours and the distance traveled was 11km. Elevator transitions are shown as vertical blue lines. The view is orthographic and the vertical axis has been exaggerated to make it easier to see each floor. The 2nd floor is approximately 90m across. Right: Floor plans for each of the floors that were mapped.

The primary contribution of this paper is an approach, called the reduced pose graph, that addresses the temporal scalability of traditional pose graphs. For long-term mapping, the size of the optimization problem should be bounded by the size of the explored environment and be independent of the operation time. To achieve this goal, the reduced pose graph reuses already existing poses in previously mapped areas, keeping the number of poses bounded by the size of the explored environment. In addition, our key insight is that new measurements can be used to further improve the map, by converting them into supplementary constraints between existing poses. The process is fluid, with new poses being added when new spaces are explored.

The advantages of the reduced pose graph extend beyond scalability. Our approach maintains multiple constraints between each pair of poses. While these constraints could be combined immediately, retaining redundancy allows for consistency checking and the detection of faulty constraints. In combination with a robust estimator this can limit the effect of erroneous constraints on the state estimation. When consensus is reached over a significant number of constraints they can eventually be combined into a single reliable constraint, thus avoiding the incorporation of erroneous measurements into the resultant pose graph edge.

Our secondary contribution is a full 6-DOF visual SLAM system which we use to illustrate and evaluate the proposed reduced pose graph formulation, as shown in Fig. 1. Our system is stereo-vision-based and operates in real-time and has been tested with data from multiple robotic platforms. A visual odometry model produces incremental constraints between key-frames which are used as input to the reduced pose graph. A place recognition module uses appearance-based methods to propose loop closures for which a geometric consistency check provides the actual constraint if successful. Place recognition allows mapping over multiple sessions, and provides improved robustness in the case of localization failure.

Robustness is further improved by using other sources of egomotion. In our work we have also utilized wheel odometry and an IMU. An accelerometer is particularly useful to eliminate drift in inclination which can accumulate in explorations as large as those presented here.

To allow for operation in large multi-floor indoor environments, our SLAM system automatically detects elevator transitions. Using an accelerometer, this approach detects characteristic elevator motion to track the vertical displacement of the robot.

We have evaluated our approach on data recorded with a PR2 mobile robot from Willow Garage. We use nine hours of data corresponding to eleven kilometers of robot trajectory recorded over a six month period, demonstrating robustness to changes in the environment. The PR2 has both a stereo camera and a RGB-D camera (the Microsoft Kinect). The main results presented here were generated using the stereo camera, while the system also supports the RGB-D camera.

II. Related work

The pose graph optimization approach to SLAM was first introduced by Lu and Milios [18] and further developed by many researchers including [9, 6, 3, 19]. Significant research has focused on providing efficient solutions, both approximate and exact. Notable examples include hierarchical representations [8], collections of local maps [5] as well as relative and non-Euclidean [23]. However few have addressed reducing the growth in size of the number of pose graph nodes as a function of time.

The visual maps by Konolige and Bowman [16] are closely related to our work. They create a skeleton graph of views similar to a pose graph. To keep the density of views or poses constant in a given region, least-recently used views are removed from the skeleton by marginalization. Our work, in contrast, avoids marginalization by not adding redundant views to begin with.

Compact pose SLAM by Ila et al. [11] uses an information-theoretic method to decide which constraints should be added. New poses are only added to the estimator (an information filter) if no other poses are nearby, while taking into account information gain from potential loop closures. The paper does not address how to limit growth when continuously operating in the same environment. In contrast, our approach can connect constraints to existing poses in areas already mapped — so as to avoid the need for the periodic addition of new nodes along the trajectory.

Kretzschmar et al. [17] also use an information-theoretic approach to decide which laser scan should be removed from the graph. They have shown large reductions in complexity for laser-based 2D pose graphs, using an approximate marginalization to retain the sparsity of the solution.

Also related to this are the sample-based maps by Biber and Duckett [1]. An initial map is created with traditional SLAM methods, and then updated at multiple different time scales to capture dynamic changes in the environment. The map is represented by a set of evolving grid-based local maps connected to an underlying pose graph. They demonstrate long-term laser based mapping on several hours of data recorded over five weeks.

For monocular SLAM, a different approach without pose graphs has been taken for managing complexity when repeatedly mapping the same environment. Most notably, Klein and Murray [13, 14] introduced monocular parallel tracking and mapping (PTAM), where a map of sparse features is updated over time by bundle adjustment, and the camera is continuously localized based on this map. Targeting augmented reality applications, the original formulation of PTAM was limited to small scale environments, mostly because of the complexity of bundle adjustment.

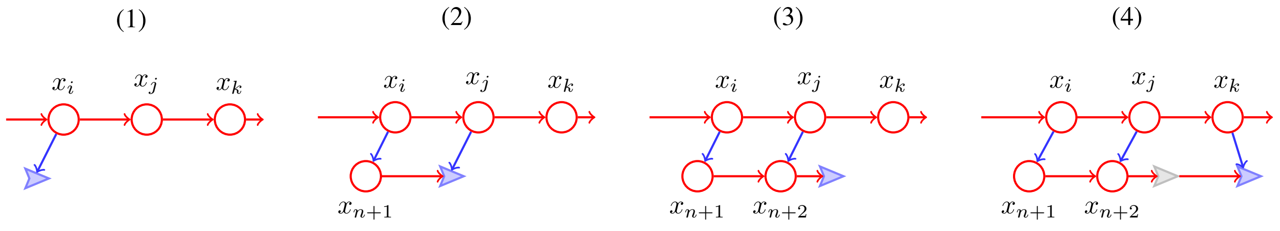
An extension to augmented reality applications to large-scale environments using several local PTAM maps is demonstrated by Castle et al. [2]. More recently, Pirker et al. [21] proposed larger scale monocular reconstruction again based on bundle adjustment. Their system updates the map in dynamic environments and achieves real-time performance with the exception of loop closures.

Eade et al. [4] reduces complexity by marginalization and degree thresholding for monocular SLAM with odometry. When the degree of a node exceeds a specific threshold, the constraint with the least residual error is removed. While suitable for low-power platforms, the estimate will be biased towards measurements with larger errors.

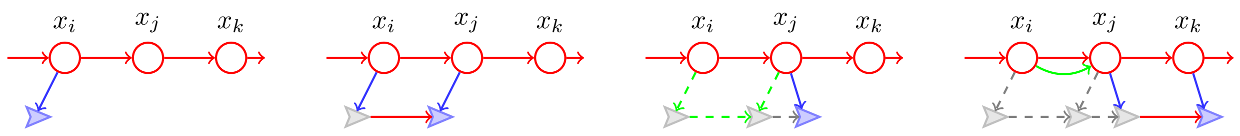
III. Reduced pose graphs

A pose graph exploits the fact that a map of the environment can be generated from a set of localized robot poses and their sensor measurements. Select poses along the robot trajectory are represented by nodes in the pose graph, typically created at fixed time intervals or after moving for a certain distance.

Independent of the application domain, constraints between consecutive poses are added based on incremental odometry (laser, wheel, visual or sonar) while loop closing constraints are found between pairs of poses (based on laser-scan, feature or visual matching). For this set of constraints, optimization finds the best configuration for all poses and is often formulated as a maximum likelihood problem. An explicit map can be formed by projecting the sensor



(a) Full pose graph.



(b) Reduced pose graph.

Fig. 2. A comparison of the construction of a full vs. reduced pose graph. The red circles represent existing poses in the map, while the blue darts represent recent poses that are not part of the map. **(a) Full pose graph**: (1) A loop closure to map pose x_i is detected. (2) The new pose (x_{n+1}) has been added to the map, and a new loop closure to x_j detected. (3) Again the new pose (x_{n+2}) has been added. (4) The same procedure is repeated for each loop closure. **(b) Reduced pose graph**: (1) A loop closure to map pose x_i is detected. (2) Because the current pose is close enough to the existing pose x_i, no new pose has been added. A new loop closure to x_j is detected. (3) A second loop closure to x_j is detected. Now the chain of constraints (dashed green) can be compounded into a single constraint, and the sequential constraint between the last two poses is dropped. (4) The new constraint (green link) has been added. A new loop closure to x_k is detected, possibly allowing creation of a constraint between x_j and x_k.

measurements at each pose into a common reference frame. By construction, this basic pose graph formulation has a significant drawback: the graph will grow unbounded with time as new poses are constantly being added.

The motivation for the reduced pose graph is to re-use existing poses in previously mapped areas. The concept of pose graph reduction to achieve temporal scalability is intuitively appealing, and indeed has been proposed in the previous literature in several contexts [7, 25, 26].

Our approach estimates the poses of a collection of keyframes as is done in PTAM [13]. Similar to FrameSLAM [15], we reduce the problem to a pose graph instead of solving the full bundle adjustment problem as is done in PTAM. In addition we continually incorporate new information as places are revisited and gradually improve accuracy over time.

This strategy corresponds to the approach taken by the exactly sparse extended information filter (ESEIF) [25, 26] to maintain sparseness and preserve consistency in an information filter context. A related and similar strategy has been previously adopted in a pose graph context by Grisetti *et al.* [7]. The challenges in implementing such an approach include deciding when to add new poses, how to minimize the loss of information, how to avoid inconsistency and how to achieve robust performance for long-term operation.

In Fig. 2 we illustrate the reduction technique by comparing the construction of a full pose graph and a reduced pose graph in the top and bottom figure respectively. The blue dart indicates the current position of the robot. In step (1) a single loop closure to the map has been acquired — as indicated by the blue edge. In step (2) a node is added to the graph so the loop closure can be used and a loop closure to x_j is detected. In step (3) a node has been added so the last loop closure can be incorporated into the graph. No loop closure is considered at the current pose because the robot has not moved far enough from the previous place. Finally in step (4) the robot has traveled enough distance that a new pose is added and the whole process is repeated.

In contrast the reduced pose graph algorithm will create a derived constraint between map poses x_i to x_j without adding new nodes. So in step (2) the transformation to x_i is updated instead of adding a new node to graph. In step (3) a complete chain from x_i to x_j (shown in green) has been formed and can now be added as a constraint to the graph. Furthermore the robot continuously tracks its position relative to the active node x_j. Then in step (4) the process is repeated as other places are visited.

A. Formulation

The reduced pose graph consists of N pose nodes $X = \{x_i\}_{i=1}^N$ and M constraints $Z = \{z_k\}_{k=1}^M$. A pose node contains the actual pose as well as the sensor measurement required to construct a map and to recognize when the pose is revisited. A constraint z_k measures the spatial relationship between two poses i_k and j_k. Because the measurements are noisy we consider the probability distribution $p(X, Z)$, in particular we are interested in the maximum likelihood solution of $p(X|Z)$ were X are our parameters of interest. By design, the joint distribution factorizes as follows

$$p(X, Z) = \prod_{k=1}^{M} p(z_k|X_k) \prod_{i=1}^{N} p(x_i), \tag{1}$$

where X_k is some subset of X. Then we can compute the maximum likelihood estimator X^* by

$$X^* = \operatorname*{argmax}_X p(X|Z) = \operatorname*{argmax}_X \prod_{k=1}^{M} p(z_k|X_k) \tag{2}$$

$$= \operatorname*{argmin}_X \sum_{k=1}^{M} - \ln p(z_k|X_k) \tag{3}$$

Algorithm 1: Reduced Pose Graph Mapping

```
Data: $map$ the map estimate
Input: $matches$ a queue for incoming matches
foreach $match \in matches$ do
    if match sucessful then
        update $\Delta_t^a$ using $match$
    else
        update $\Delta_t^a$ using wheel odometry
    extract descriptor from current frame
    check for loop closure to active node
    if not found then
        check for a global loop closure
    if found then
        if found node is active node then
            update transform to active node
        else
            add transformation to found node
            set found node as active node
    else
        if in a new place then
            add a node for new place
            set new node as active node
update $map$
```

and for distributions of the form $p(z_k|X_k) \propto e^{C_k(f(X_k)-z_k)}$ we get the following nonlinear optimization problem

$$X^* = \underset{X}{\operatorname{argmin}} \sum_k C_k \left(f(x_{i_k}, x_{j_k}) - z_k \right), \tag{4}$$

where f is a function that predicts a constraint from two poses and C_k is a cost function associated with constraint k. The same solution applies to the traditional pose graph with the only difference being the way that constraints and nodes are created. For a Gaussian error distribution the cost function is the Mahalanobis distance.

B. Graph Construction

Let us now consider how the reduced pose graph is constructed; a summary is provided in Algorithm 1. The graph is initialized by defining the first pose as the origin. There are three primary operations performed: tracking, adding nodes, and adding loop closures. Let x_a denote the active pose and Δ_t^a denote the transformation from the active node to the current position at time t.

1) Tracking: The tracker estimates the distribution $p(\Delta_t^a|U_t, Z_t)$ where U_t and Z_t are the odometry and visual constraints respectively. Under our Markov assumption the distribution can be computed recursively

$$p(\Delta_t^a|U_t, Z_t) \propto p(\Delta_t^a|u_t, z_t, \Delta_{t-1}^a) p(\Delta_{t-1}^a|U_{t-1}, Z_{t-1}). \tag{5}$$

In practice we track the location distribution by compounding with the new measurement at each stage. The compounding of uncertain transformations follows the notation of Smith et al. [24]. Given a chain of transformations $\{z_{12}, z_{23}\}$ with covariances $\{\Sigma_{12}, \Sigma_{23}\}$ respectively, the compounded transformation z_{13} is computed as follows

$$z_{13} = z_{12} \oplus z_{23} \tag{6}$$
$$\Sigma_{13} = J_{1\oplus}\Sigma_{12}J_{1\oplus}^T + J_{2\oplus}\Sigma_{23}J_{2\oplus}^T, \tag{7}$$

where $J_{1\oplus}$ and $J_{2\oplus}$ are the Jacobians of the compound operation $\oplus$ with respect to z_{12} and z_{23} respectively. This is the first order approximation of the mean and covariances, where z_{12} and z_{23} are assumed to be independent. The factor added to the graph will then be $|f(x_1, x_3) - z_{13}|_{\Sigma_{13}}$, where f is a function that computes the transformation between x_1 and x_3, i.e. $x_3 = x_1 \oplus f(x_1, x_3)$.

2) Adding nodes: A new node is added if the currently estimated position, $x_a \oplus \Delta_t^a$, is in a location where there is no existing node. Then a new node x_{N+1} is added to the graph and the new distribution is computed by

$$p(X_{N+1}, Z_{M+1}) \propto p(z_{M+1}|x_a, \Delta_t^a) p(X_N, Z_M). \tag{8}$$

The newly added node becomes the new active node x_a and the tracker is re-initialized.

3) Adding loop closures: The tracker is always trying to find an alignment with an existing pose. First the node that is closest to the current position is tested. If that fails a global appearance based search is used to discover a loop closure. If successful, a new constraint is added to the graph. The registration is given by

$$p(\Delta_t^b|x_a, \Delta_t^a, x_b), \tag{9}$$

where x_b is the node being registered to. After the loop closure is added x_b becomes the active node. Next the robot needs to re-localize relative to the map to avoid reusing the previous measurement. A re-localization will initialize the active node and the relative position to it. Alternatively, if enough inliers were found in the loop closure, they could be split into two sets, one for the loop closure and the other for re-localizing. In comparison a full pose graph approach would have added a new pose at the point of loop closure.

In case x_a and x_b are the same node it is possible to choose if the current estimate or this loop closure should be used as the estimate of the position relative to x_a by considering which has higher entropy, using the test $det(\Sigma_t^a) > det(\Sigma_t^b)$.

The graph is sparsified by discarding some of the sequential constraints and then marginalizing along a chain of poses. The algorithm continuously incorporates new information as places are revisisted. This gradually improves the accuracy of the map as shown in Fig .3. Most of the updates are additions of new constraints (but not variables) to the estimation problem, which is applied incrementally in an efficient manner using the iSAM algorithm [12] for pose graph optimization, which also includes an incremental implementation of the Powell's dog leg algorithm [22] for use with robust estimators.

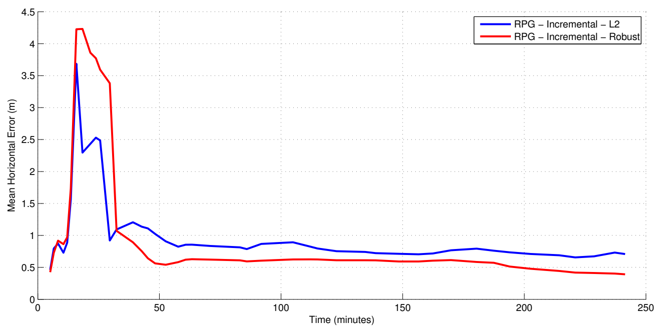


Fig. 3. Estimation error for a subset of the poses over time. The blue and red lines are the mean error when using squared and pseudo-Huber cost respectively. The estimation error is computed as the Euclidean distance, in the XY plane, between the estimated position and the groundtruth.

IV. Visual SLAM

In this section we demonstrate the reduced pose graph concept in a complete visual SLAM system. The main components are: visual odometry [10], loop proposal using bag of visual words, and map state estimation using iSAM. In typical operation the visual odometry algorithm is run on a single thread at 30Hz (the camera framerate) while the remaining components are executed on a second thread.

A. *Visual Odometry*

Incremental motion is estimated using an efficient implementation of stereo-visual odometry. This approach, named FOVIS (Fast Odometry for VISion) is explained and quantified in more detail in Huang et al. [10]. The output from the visual odometry are a pair of matched frames. The match includes an estimated 6-DOF transformation between the frames, detected keypoints, and correspondences between the keypoints. The FOVIS library supports both stereo cameras and RGB-D sensors (such as the Microsoft Kinect) and results for both camera types are presented in Section V.

Additionally, our approach incorporates IMU (roll and pitch) and wheel odometry (horizontal translation) in situations in which the vision system cannot estimate motion, e.g. featureless walls, low light, and occlusion of the camera. If required, it is also possible to use the wheel odometry (WO) as the sole source of relative motion estimation — using vision data only for the detection of loop closures. Results comparing these different options is given in Table I. In section IV-F we show how the IMU can be used to detect floor transitions.

	Median (m)	Mean (m)	StdDev (m)
Vision only	0.74	0.73	0.55
WO + IMU	0.71	0.79	0.52
Vision + IMU + WO	0.59	0.58	0.38

TABLE I
ACCURACY FOR DIFFERENT VARIANTS OF THE SLAM ALGORITHM. IN ALL CASES THE CAMERA IMAGES ARE USED FOR LOOP CLOSURES.

B. *Map Management*

The map management module receives incoming measurements from the visual odometry, IMU and other sensor inputs. For any incoming frame a feature descriptor is computed for each keypoint in the new frame. The feature descriptors are maintained for later use in the appearance-based loop proposal and frame registration modules. Several different descriptor types are supported by our implementation including BRIEF, Calonder and SURF. In each case we utilize their OpenCV implementations.

To limit of the number of poses estimated the explored space is partitioned. Our current implementation partitions each floor according to the robot's 2D motion (i.e. in x, y, and heading). The partitioning is updated each time the map estimate is updated. An extension of this approach to full 3D motion can be based around the volume observed from a particular camera pose. If multiple nodes are assigned to the same grid cell, the node most recently added is selected as the active node. This map is then used to actively localize the robot by continuously registering the current image frame against this active node.

C. *Active Node Registration*

To register two frames to one another a collection of putative matches are found by matching each keypoint to the keypoint that has the closest feature descriptor. This is done using brute-force matching. The descriptor we most commonly use is the BRIEF descriptor, which can be compared very efficiently. Inliers are determined by finding a maximal clique in a graph of consistent feature pairs [10]. Finally the 6-DOF transformation between the two frames is estimated by minimizing the bi-directional re-projection errors of these matched keypoints. If the number of inliers is within a threshold the registration is accepted.

D. *Global Node Registration*

If registration to the active node fails, a global loop closure algorithm (across the entire set of poses) is used instead. A bag of visual words approach is used to describe each frame and using an inverted index an efficient search is carried out for similar frames. We use Dynamic Bag-Of-Words [20] for SURF descriptors and Binary Bag-of-Words (BBoW) [1] for BRIEF descriptors. Each match is scored and if the score is below a given threshold the frame is proposed as a possible loop closure. A typical distribution of the scores is shown in Fig. 5. By setting an appropriate threshold on the score we can choose between having many false proposals and missing potential matches. The loop proposal is then verified with a geometric consistency check, by registering the two frames, using the method described above.

E. *Map Estimation*

The result of both the visual odometry and visual registration algorithms are inserted into the SLAM map estimation algorithm which is based on iSAM [12]. This approach effectively solves the non-linear least squares problem induced by the pose graph, albeit in an efficient incremental manner. An

[1]The library used for the binary descriptors is called Binary Bag-Of-Word (BBoW) developed by Daniel Maturana — `http://dimatura.net/proj_bbow.html`

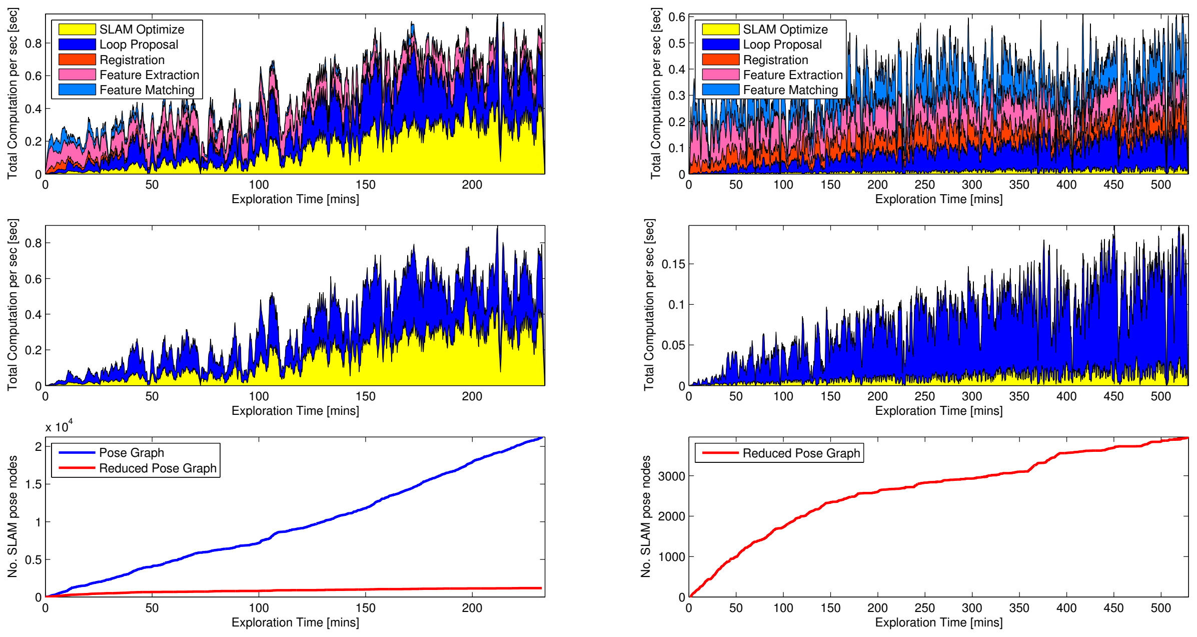
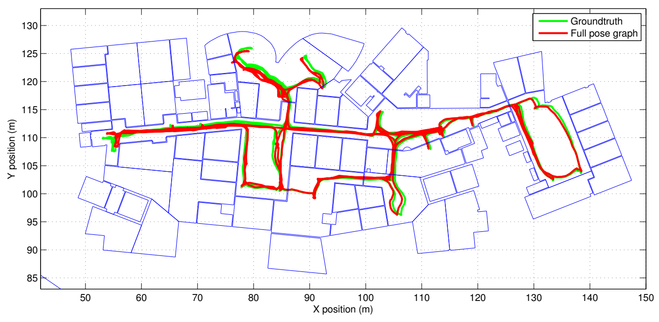
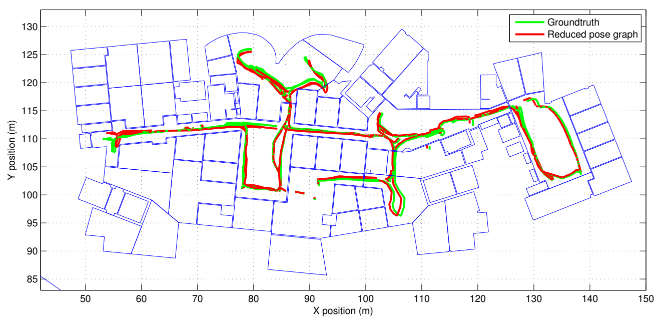


Fig. 7. Timing results when using a **full pose graph** (left) for a 4 hour sequence and a **reduced pose graph** (right) for a 9 hour sequence (i.e. 5 additional hours). The top row of plots shows the time for each component of the SLAM system as a function of exploration time. The middle row shows only the components that have growing time complexity. The lower row shows the number of nodes in the graph — both for the pose graph (left) and the reduced pose graph (both). As detailed in Section V the core system infrastructure was specifically designed to maintain real-time operation by delaying optimization if necessary. After 4 hours the full pose graph spends a majority of the available computation time doing so, resulting in delayed optimization and few loop closure constraints being found. Meanwhile the reduced pose graph approach remains comfortably real-time.



(a) Pose graph – 28520 poses



(b) Reduced pose graph – 1363 poses

Fig. 4. A comparison of a full pose graph vs. a reduced pose graph from 4 hours of traversal. The green is the groundtruth trajectory and red are the estimated trajectories. The average error for the full pose graph is 0.43m while it is 0.47m for the reduced pose graph.

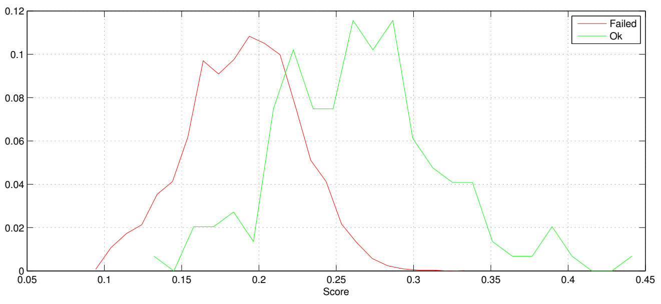


Fig. 5. The distribution of the failed and successful loop proposals as a function of the score from the system.

update is carried out each time new information is added to the graph: either a new node or a constraint between existing nodes. By utilizing the reduced pose graph approach, the rate at which the problem grows is reduced significantly, as demonstrated in Fig. 7.

F. Vertical Motion: Elevators

For many multi-floor buildings it is reasonable to assume that the robot can go from one floor to another using an elevator. This type of motion is not observable by the vision system or the wheel odometry. Also it is not sufficient to rely on intent only, because the robot does not have full control over which floors the elevator will stop at.

To detect elevator rides we track the vertical motion using an accelerometer sensor. Integrating the vertical accelerometer information over time results in the vertical displacement. The method is accurate because the velocity at the start and

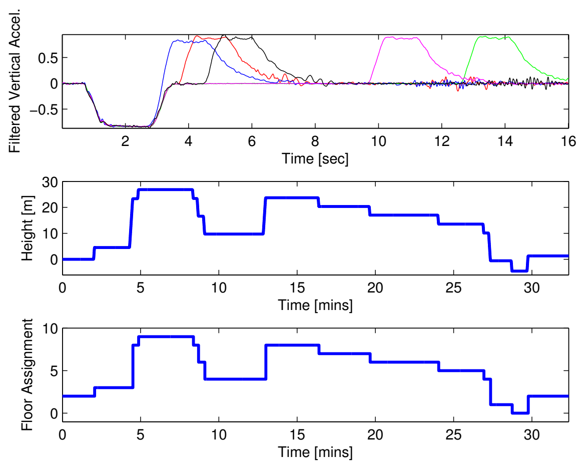


Fig. 6. Top: Using the Z-component of the PR2's accelerometer, the start and end of elevator transitions can be detected using a matched filter. The figure illustrates 5 different elevator rides - showing that the acceleration is clearly repeated. Middle: By integrating this signal the elevation of each floor can be estimated. Bottom: During our 10 floor experiment (beginning and ending on floor 3), the floor assignment can be determined using a simple lookup table. See Section IV-F for more details.

TABLE II

APPROXIMATE FIGURES OF INTEREST CORRESPONDING TO THE 10 FLOOR EXPERIMENT ILLUSTRATED IN FIG. 1.

Duration of Experiment	6 months
Operation time	9 hours
Distance Traveled	11km
VO keyframes	630K
Failed VO frames	87K
Registrations	303K
Loop proposals	30K

end of the elevator transit are known to be zero. Results from our floor tracker are show in Fig. 6.

To assign these vertical displacements to floor numbers, a table of floor heights is maintained. Each time a transition is detected the table is searched for the closest floor. If the distance to that floor is within a given threshold it is accepted as the current floor, otherwise a new floor is added to the table. Knowing the floor the robot is on is also useful for limiting loop closure search in order to avoid wrong matches based on self-similarities between floors.

V. RESULTS

We evaluated the system with the MIT Stata Center vision data set[2] (see Table II) that was collected by manually driving a PR2 through the building. The PR2 was equipped with a stereo camera, a Kinect sensor and a Microstrain IMU among other sensors. The data was collected over a period of six months. There were repeated excursions through one of the floors (see Fig. 8) and occasional visits to the other floors. In total 10 floors were visited; a map spanning all the floors was created (see Fig. 1).

As shown in Fig. 7, the rate at which nodes are added to the graph reduces as time progresses. When new areas were first explored this rate increased as each new place was recorded. The loop proposal and the optimization modules grow in complexity as more nodes are added to the graph. However, as shown in Fig. 7 these modules account for only a small fraction of the total running time of the system and are much reduced when compared to the traditional full pose graph. The majority of computation time is spent on frame registration and feature extraction, both of which are constant time. Visual odometry runs at 30Hz on a separate thread, and loop closure runs at 2Hz.

To compare the full pose graph with the reduced pose graph we used a 9 hour data set collected over 6 months and encompassing 11km of motion. After just 4 hours the full pose graph approach uses a majority of the available computation time optimizing the graph (See Fig. 7). Doing so will eventually affect the localization accuracy of the system. This happens because the core system infrastructure was developed to adaptively balance the load so that it can meet real-time requirements when running on a robot. This is achieved by having the mapping thread consider a limited number of loop closure proposals and also adapting how often the map optimization is executed. Thus, as the pose graph grows, the system cannot keep up and fewer and fewer frames are registered with the map. This figure also illustrates that the reduced pose graph does not suffer this problem and after 9 hours continues to expand and improve its map without sacrificing map accuracy.

An overview of pose estimates from the two two methods is provided in Fig. 4. The reduced pose graph is an order of magnitude smaller than the full pose graph, yet they both represent the environment to similar accuracy. Here the data was played back at a slower rate so the full pose graph would not be adversely affected by the increased computation time. We have compared these results to ground truth obtained by manually aligning laser scans to an architectural floor plan. The error for the full pose graph is 0.43m mean absolute error with 0.27m standard deviation, while the reduced pose graph has 0.46m mean absolute error with 0.29m standard deviation.

Finally to demonstrate the versatility of our approach, in Fig. 8 we present a reduced pose graph produced using RGB-D from the robot's Microsoft Kinect. It can be seen that the map closely resembles the corresponding floor plan.

One of the lessons learned from this experiment was that it is essential to incorporate more than one sensor input. In our case we used wheel odometry and the IMU. The visual odometry typically failed in the elevators, when going through areas where the lights had been turned off, when turning around corners while looking at featureless walls, and because of people obstructing significant portions of the camera view. We have also observed that by continually incorporating information into the estimate the robot was able to correct the map in later passes if failures had occurred during a previous visit. The conclusion is that these two aspects are important to robust operations.

[2]More information about the MIT Stata Center Mapping Data Set is available at `http://projects.csail.mit.edu/stata/`

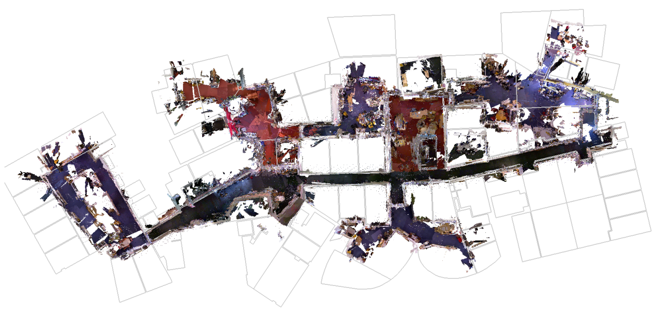


Fig. 8. Map created using a Kinect camera. The map is aligned using the best similarity transform that fits the pose graph to the ground truth.

VI. Conclusion

This paper proposes a reduced pose graph formulation which enables large scale mapping over long time durations. In an environment which has been previously explored, this approach adds extra pose-to-pose constraints instead of adding new and redundant poses to the underlying pose graph. This important modification allows the SLAM system to scale in size with area of exploration instead of the time of exploration. Additionally these constraints enable continual improvement of the map solution as inconsistent constraints can be pruned.

The algorithm was demonstrated within a visual SLAM system and its effectiveness was demonstrated on a large data set where the same environment was frequently re-visited by an exploring robot. In addition to stereo-vision and RGB-D, information from sensors such as a robot's wheel odometry and IMU can also be incorporated and result in improved robustness in situations where the vision-only system would fail. We have also shown how elevator transits can be detected — enabling seamless multi-floor mapping.

There are still several issues that remain to be explored within the proposed model. Our current implementation cannot fully guarantee that the graph will not grow with time. When the robot becomes temporarily disconnected from the map, it will add poses to the graph until re-localized. Changes in the environment can also result in new nodes being added. We believe that this issue can be handled by introspection of the overlapping poses when re-localization finally occurs.

References

[1] P. Biber and T. Duckett, "Experimental analysis of sample-based maps for long-term SLAM," *Intl. J. of Robotics Research*, vol. 28, no. 1, pp. 20–33, 2009.

[2] R. Castle, G. Klein, and D. Murray, "Wide-area augmented reality using camera tracking and mapping in multiple regions," *Computer Vision and Image Understanding*, vol. 115, no. 6, pp. 854 – 867, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1077314211000701

[3] F. Dellaert, "Square Root SAM: Simultaneous location and mapping via square root information smoothing," in *Robotics: Science and Systems (RSS)*, Cambridge, MA, 2005.

[4] E. Eade, P. Fong, and M. Munich, "Monocular graph SLAM with complexity reduction," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Oct. 2010, pp. 3017–3024.

[5] C. Estrada, J. Neira, and J. Tardós, "Hierarchical SLAM: Real-time accurate mapping of large environments," *IEEE Trans. Robotics*, vol. 21, no. 4, pp. 588–596, 2005.

[6] J. Folkesson and H. Christensen, "Graphical SLAM - a self-correcting map," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, vol. 1, 2004, pp. 383–390.

[7] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard, "A tree parameterization for efficiently computing maximum likelihood maps using gradient descent," in *Robotics: Science and Systems (RSS)*, Jun. 2007.

[8] G. Grisetti, R. Kümmerle, C. Stachniss, U. Frese, and C. Hertzberg, "Hierarchical optimization on manifolds for online 2D and 3D mapping," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Anchorage, Alaska, May 2010.

[9] J. Gutmann and K. Konolige, "Incremental mapping of large cyclic environments," in *Proc. of the 1999 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, vol. 1, 1999, pp. 318–325.

[10] A. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an RGB-D camera," in *Proc. of the Intl. Symp. of Robotics Research (ISRR)*, Flagstaff, USA, Aug. 2011.

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

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

[13] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *IEEE and ACM Intl. Sym. on Mixed and Augmented Reality (ISMAR)*, Nara, Japan, Nov. 2007, pp. 225–234.

[14] ——, "Improving the agility of keyframe-based SLAM," in *ECCV '08: Proceedings of the 10th European Conference on Computer Vision*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 802–815.

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

[16] K. Konolige and J. Bowman, "Towards lifelong visual maps," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Oct. 2009, pp. 1156–1163.

[17] H. Kretzschmar, C. Stachniss, and G. Grisetti, "Efficient information-theoretic graph pruning for graph-based SLAM with laser range finders," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Sep. 2011.

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

[19] E. Olson, J. Leonard, and S. Teller, "Fast iterative alignment of pose graphs with poor initial estimates," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2006, pp. 2262–2269.

[20] P. Pinies, L. Paz, D. Galvez-Lopez, and J. Tardos, "CI-Graph SLAM for 3D reconstruction of large and complex environments using a multicamera system," *J. of Field Robotics*, vol. 27, no. 5, pp. 561–586, Sep. 2010.

[21] K. Pirker, M. Ruether, and H. Bischof, "CD SLAM - continuous localization and mapping in a dynamic world," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Sep. 2011.

[22] D. Rosen, M. Kaess, and J. Leonard, "An incremental trust-region method for robust online sparse least-squares estimation," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, St. Paul, MN, May 2012, pp. 1262–1269.

[23] G. Sibley, C. Mei, I. Reid, and P. Newman, "Planes, trains and automobiles – autonomy for the modern robot," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. IEEE, 2010, pp. 285–292.

[24] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," in *Autonomous Robot Vehicles*. Springer Verlag, 1990, pp. 167–193.

[25] M. Walter, R. Eustice, and J. Leonard, "Exactly sparse extended information filters for feature-based SLAM," *Intl. J. of Robotics Research*, vol. 26, no. 4, pp. 335–359, 2007.

[26] Z. Wang, S. Huang, and G. Dissanayake, "D-SLAM: A decoupled solution to simultaneous localization and mapping," *Intl. J. of Robotics Research*, vol. 26, no. 2, pp. 187–204, 2007.