A Framework for Multi-Robot Pose Graph SLAM

Isaac Deutsch[1], Ming Liu[2] and Roland Siegwart[3]

***Abstract*— We introduce a software framework for real-time multi-robot collaborative SLAM. Rather than building a complete SLAM system, our framework is designed to enable collaborative mapping for existing (single-robot) SLAM systems in a convenient fashion.**

The framework aggregates local pose graphs obtained from its multiple robots into a global pose graph, which it then feeds back to the robots to increase their mapping and localization effectiveness. The framework can potentially work with various SLAM algorithms, as long as they provide a pose graph with an image associated with each node, and absolute scaling. The merging of pose graphs is purely visual-based and does not require well-defined initial robot positions nor environment markers.

To handle network delays, we propose a graph correction scheme that avoids using mutexes (and thus avoids modifying the existing SLAM system) by assuming local graph consistency. Furthermore, we propose a simple image feature filtering method that uses an associated depth image to filter image features unsuitable for scene recognition.

We demonstrate the framework's functionality with several interior datasets that we have collected using three robots.

I. INTRODUCTION

Simultaneous Localization and Mapping (SLAM) appears in many robotics-related challenges such as self-driving cars, crop surveillance, and rescue missions. It attempts to solve the circular problem of localizing a robot using a map of its environment while simultaneously generating that map.

One broad category of SLAM algorithms are the approaches that build and maintain a pose (position and orientation) graph, consisting of robot states (as nodes), and pose transformations (as edges, i.e., constraints). Each robot state contains the pose and sensor data (e.g., camera images or laser scans). The drifting error from odometry can be reduced by detecting re-visited places and introducing loop closure constraints, then using global graph optimization. The goal of graph optimization in this context is to find a configuration that minimizes the error of the constrained pose estimations[1]. In Fig. 4, the error minimization is illustrated in the example of a loop closure. The map can be reconstructed at any time from the graph by superimposing the associated sensor data and finding a consistent configuration.

[1,3]Isaac Deutsch and Roland Siegwart are with the Autonomous Systems Lab, ETH Zurich, Switzerland – {deutschi,rsiegwart}@ethz.ch

[2]Ming Liu is with the Robotics and Multi-perception Lab (RAM-LAB), Dept. of Mechanical and Biomedical Engineering, City University of Hong Kong, Hong Kong – mingliu@cityu.edu.hk

This work was supported by the Research Grant Council of Hong Kong SAR Government, China, under project No. 16206014 and No. 16212815; National Natural Science Foundation of China No. 6140021318, awarded to Prof. Ming Liu

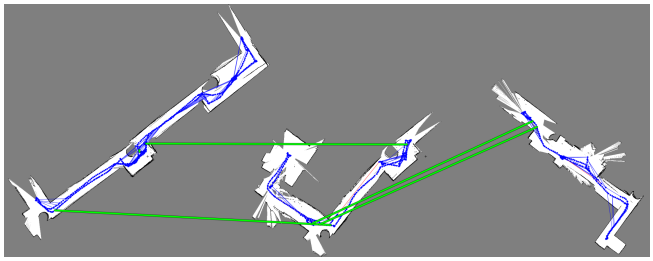

Fig. 1. The three sub-maps and their connections as generated by our framework ("office" dataset).

As the graph is a sparse representation of the whole SLAM state, graph-based approaches are well-suited for large-scale applications[2] and for collaborative SLAM. Using a multi-robot system for these applications is promising because the SLAM problem can be distributed[3]. A team of robot explorers can split up at a pathway fork and later meet again, then share and merge their maps. Another advantage of multi-robot systems is data redundancy in overlapping areas, which enables robots to "help each other out" in case of localization loss. In various environments, one can profit from heterogenous robots for mapping. For example, a multi-story building could be mapped by a flying drone in collaboration with a second driving robot. To combine their respective mapping results, a comprehensive framework should be able to process data from different SLAM systems (e.g., monocular and laser scanner SLAM).

Therefore, a multi-robot SLAM framework that aims to combine individual SLAM results must have at least two primary functions to tackle the aforementioned problems: *1)* It is necessary to have an algorithm that can detect when robots meet or when a place has been visited multiple times (also known as *place* or *scene recognition*). In case of a detection, the framework must be able to align the pose graphs from both robots. *2)* It is necessary to have a communication system that allows the robots to share information, either directly or over a server.

We propose a framework for teams of robots doing SLAM, called *Team SLAM* (or *TSLAM* for short). The novelty of the framework is that it can combine existing single-robot SLAM software into a multi-robot SLAM application. Our goal was to make TSLAM developer-friendly by making it easy to setup and to interface.

TSLAM is capable of combining multiple pose graphs from distributed robots by means of visual scene recognition, visual relative pose calculation, and graph optimization. The framework consists of two parts: One part runs on a central workstation computer and is responsible for the aggregation

of data and for giving feedback. The second part runs on the mobile robots and is responsible for image pre-processing, sending pose graph updates, and receiving feedback. The feedback includes data from all robots. This enables indirect inter-robot communication (e.g., of mapping results).

To handle delays in data transmission over the network, usually it is necessary to use mutual exclusion (mutex) locks on all affected pose graphs whenever a global optimization is performed to avoid data corruption. For existing SLAM systems, adding such a locking functionality can require large changes in the system structure. For this reason, we propose an algorithm which can detect and automatically correct for data corruption caused by network delays, which eliminates the need for mutexes.

In TSLAM, the alignment of pose graphs from different robots is based on feature matching. Thus, the accuracy of the graph alignment depends on the quality of the features. We propose a feature filtering method, which can be employed when using images accompanied by range information such as those coming from RGB-D cameras, stereo cameras, or camera and laser scanner combinations.

The remainder of this paper is structured as follows: Section II lists related research. Section III particularizes the posed problems and explains the methods we use to solve them in more detail. In sections IV and V, we present experiment results and discussion. Finally, we summarize the achievements of our research in section VI.

II. RELATED WORK

Multi-robot collaborative SLAM is not a new concept and has been widely researched before. An overview is given here to motivate which components we chose to use as part of our framework and to see if there has been other work on enabling existing single-robot SLAM systems to perform collaborative SLAM.

A. Graph-based SLAM

An overview of graph-based SLAM methods is given by [4]. Examples of recent single-robot graph-based SLAM systems include ORB-SLAM[5] and iSAM2[6].

Recently, algorithms that operate directly on sensor data and skip feature extraction and matching (so-called *direct* methods) have given rise to direct monocular SLAM approaches[7], [8]. The aforementioned ORB-SLAM has also been enhanced in that way[9].

B. Graph Optimization

Numerous works have specifically explored graph optimization in the context of graph-based SLAM[10]. "g2o" offers a highly specialized C++ library implementation[1]. As graph-based SLAM is not robust to false loop closures (including false scene recognition positives), [11] has developed an optimization algorithm that can effectively disable the constraint outliers. In this paper, we also explore the idea of rejecting constraint outliers to improve the handling of network delay in distributed graph optimization.

C. Scene Recognition

For scene recognition and performing loop closure in the SLAM graph, image features are commonly extracted[12], [13]. FAB-MAP implements a complete SLAM system on top a bag-of-words (BOW) approach[14]. Research such as [15] and [16] use image depth data to improve the scene recognition. In section III-C, we investigate a similar idea.

D. Multi-robot SLAM

Building a multi-robot system from scratch has been researched before, sometimes using components from preceding works[17]. A generalization from single-robot SLAM by launching multiple robots from a predefined setup includes [18]. Moving away from predefined setups, relative robot localization becomes important[19], [20]. This idea can be combined with scene recognition[21]. Several works proposed distributed computing of the separate SLAM tasks[22], [23]. Because scalability both with number of robots and with the map size is significant, various works adopt special data representations[24], [25]. [26] demonstrates a single-camera framework for micro aerial vehicles, which requires handling scale drift. All of these works achieve multi-robot SLAM by implementing a very specific SLAM framework and are not able to adopt to new developments. In this work, we show that our framework can process data from an existing single-robot SLAM system to form a new multi-robot SLAM system. This allows our framework to adopt to novel contributions from other works, and to employ suitable SLAM algorithms for heterogenous robots.

III. PROBLEM FORMULATION AND METHODS

In this section, we describe the hardware used and the different modules of the software framework.

A. Development Hardware

For development of the framework, we use a team of three robots. Each consists of a mobile wheeled base with rotary encoders, a RGB-D camera, and a netbook with Wi-Fi connectivity. The robots are remote-controlled by a joystick. The central workstation consists of a modern laptop.

B. Overview of Data Flow and Processing

The data is gathered and processed on the distributed robots by their (single-robot) SLAM system. From there, it is sent to an image pre-processing module, which passes the result to the central workstation. There, multiple steps are performed (see Fig. 2). In case a graph optimization is performed, feedback is sent back to the affected robots.

We use the Robot Operating System (ROS)[27] to handle asynchronous communication over a Wi-Fi network between the central workstation, which acts as a server, and the distributed robot clients. The robots do not need to be connected to the central workstation at all times. After disconnection, they just continue with mapping by themselves. Upon reconnection, all missing pose graph and map data is added to the global graph. However, the images recorded during disconnection will not be sent at a later time for scene

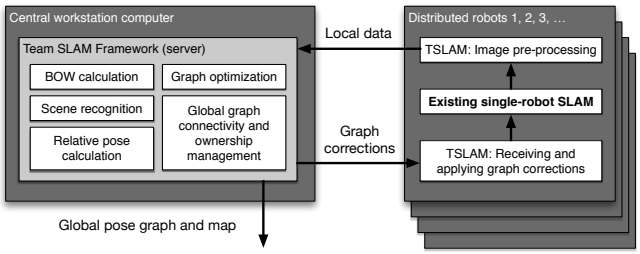

Fig. 2. Schematics of the overall system setup and framework modules.

recognition to avoid using a lot of bandwidth and impeding the real-time operation of the framework.

The interfacing that needs to be done to connect a single-robot SLAM system to TSLAM consists of two steps: *1)* Upon creation of a new node, a pose graph update message has to be delivered. It consists of the pose graph including the new node and the current camera image. The image preprocessing module will then process the message and pass it on to the central workstation. *2)* A callback function has to be implemented which accepts an incoming graph adjustment message and updates the local graph accordingly.

C. Image Pre-processing

This module runs on the distributed mobile robots. It takes a pose graph update message, filters the contained camera image, then passes the information on to the central workstation. To save bandwidth, the camera image is not sent to the central workstation. Instead, only the feature points are sent for the purpose of scene recognition and relative pose calculation. To keep this module lightweight, only the following necessary computations are performed:

1) *Images are scaled* to a certain resolution to have image features of comparable scale.
2) *ORB[28] feature detection and description.*
3) If depth information registered to the color image is available, for example through a RGB-D camera, it is used for *feature point filtering*. See below.

We have chosen ORB over other feature detectors/descriptors because it offers a good compromise of computation speed and performance[28].

For relative pose calculation, the extracted features are not optimal in some cases. We have noted three types of "bad" feature points with respect to their depth in the scene: *1)* Features corresponding to scene parts far away in the scene are undesirable because their position is noisy. For example, in a stereo camera setup, large depth in relation to the stereo baseline gives unreliable results. *2)* The depth value of features corresponding to a sharp object corner or edge can "jump" between a point close to the camera and a point far away from the camera. *3)* Feature points on curved objects are more prone to occlusion, i.e., they might be obstructed when looking from a different angle.

If depth information is available, the feature points are filtered according to the criteria above. To this end, we implement a "cross" filter, see Fig. 3. First, feature points that

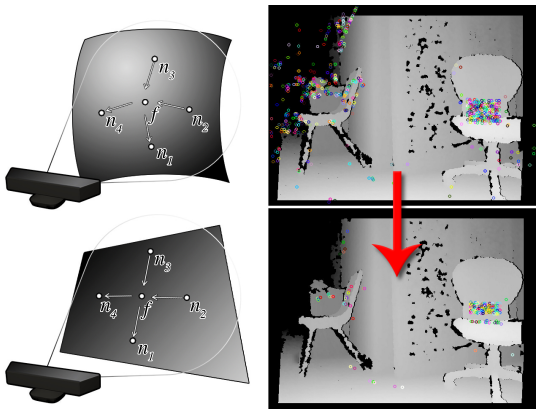

Fig. 3. Left: cross filter showing four vectors in curved and in planar case. Right: Exemplary filtering result. (Best viewed in color.)

are associated with a large depth value, or that are close to the image border are discarded. Then, four neighboring points ($n_{1,2,3,4}$ in the figure) are selected. From these neighbors, four vectors are constructed as the point difference between the neighbors and the feature point (e.g., $(f - n_3)$ and $(n_1 - f)$). If the sum of angles between each horizontal and vertical pair of vectors is greater than a certain threshold, this means that the feature point is on a non-flat surface, and it is thus discarded. A filtering result is shown in Fig. 3. While we do not have a quantitive measure of the effect on relative pose accuracy, the cross filter will remove many feature points which are effectively unusable for relative pose calculation.

D. Pose Graph Merging and Optimization

The feature points of the image pre-processing are sent to the central workstation. There, we check if a new pose graph edge can be introduced:

1) The *bag-of-words representation is extracted.*
2) *Scene recognition* is performed by comparing the new BOW data against the previously collected data. We use an open-source version of FAB-MAP[29].
3) If the probability of a scene recognition is very high, *image features in both images are matched*, and we *calculate the relative pose* between the two views. We use the eight point algorithm[30], which is motivated by section IV-A.
4) Some geometrical heuristics are checked: If the relative pose is impossible because the relative orientation is higher than the camera field-of-view or they are too far apart (in case a relative position estimate is known already), the pose graph merging is discarded. Also, subsequent views from the same robot are not considered (the individual SLAM systems will do just that on their own). Otherwise, a new constraint with the estimated relative pose is introduced in the global pose graph.

We assume the system to have access to absolute map scaling (e.g., using range measurements, IMUs, or odometry

from wheel encoders), which gives the (otherwise ambiguous) scaling to the relative pose calculation. We thus assume that these systems have a good estimate of their own graph scale (including scale drift), and therefore TSLAM combines their graphs and maps in the scale they are received.

When a new constraint is introduced because of scene recognition, nonlinear graph optimization using least-squares is performed. For that, we use g2o, which compares favorably against similar packages[1]. After global graph optimization is finished, the corrected graphs are sent to all connected robots. As we show in section V-B, this helps every robot to keep a consistent local localization and mapping estimate.

Graph optimization is also triggered by other events: when a large number of new nodes is introduced without intermediate optimization, when a low thrust value θ_{min} is found during optimization (see section III-E), or when one of the robots reconnects after a disconnect.

E. Handling Network Delays in Graph Optimization

During the graph optimization, due to the time delay of the message from the central workstation to the robot, it sometimes occurs that the client creates new nodes in continuation of the old, uncorrected node positions (see Fig. 4, (a) and (b)). This can result in skewed graphs, where a part lies in the corrected position while another part immediately adjacent to it is in the uncorrected position (Fig. 4 (c)). To resolve this while avoiding the need for mutex locks, we introduce the concept of the *trust factor*, θ. It serves to find a linear combination of pre- and post-optimization poses, rejecting inconsistent optimizations. θ is based on the assumption that edge orientations exhibit greater error than edge lengths. This assumption is justified by the fact that edge orientation is affected by the accumulated pose drift of all nodes since the last optimization, but edge length is affected only by the pose drift between the two affected nodes. Edges are updated using Eq. 1 and 2, where $\mathbf{x_1}$ and $\mathbf{x_2}$ are the two poses defining the edge, $\hat{\mathbf{x}}_\mathbf{i}$ are the optimizer outputs and $\mathbf{x_i}\prime$ are the resulting new poses:

$$\mathbf{x_1}\prime = \hat{\mathbf{x}}_\mathbf{1} \tag{1}$$

$$\mathbf{x_2}\prime = \mathbf{x_2} \cdot (1-\theta) + \hat{\mathbf{x}}_\mathbf{2} \cdot \theta \tag{2}$$

where the trust factor θ is calculated by:

$$\theta = \left(\frac{2 \cdot |\mathbf{x_2} - \mathbf{x_1}| \cdot |\hat{\mathbf{x}}_\mathbf{2} - \hat{\mathbf{x}}_\mathbf{1}|}{|\mathbf{x_2} - \mathbf{x_1}|^2 + |\hat{\mathbf{x}}_\mathbf{2} - \hat{\mathbf{x}}_\mathbf{1}|^2} \right)^3 \tag{3}$$

$\theta(a,b)$ is modelled after the following properties: $\theta(a,b) \ll 1 \Leftarrow |a-b| \gg 0$ and $\theta(a,b) = 1 \iff a = b$:

$$\theta(a,b) = \left(\frac{1}{2}(\frac{a}{b} + \frac{b}{a}) \right)^{-3} = \left(\frac{2 \cdot a \cdot b}{a^2 + b^2} \right)^3 \tag{4}$$

The effect of using θ can be seen in Fig. 4 (d), which shows a consistent solution after two optimizations.

The trust factor θ does not take any other information into account (such as the information matrix of the optimizer). Its purpose is solely the post-processing of the optimized graph, while serving as a consistency check. Since both the pre- and the post-optimization poses are valid, the linear combination will also be a valid pose. For numerical reasons, we set $\theta = 1$ for very small edge lengths.

This way, we can keep a globally consistent graph by re-running inconsistent graph optimization until a consistent result is achieved. We have found that a consistent solution is reached in most cases after no more than two iterations.

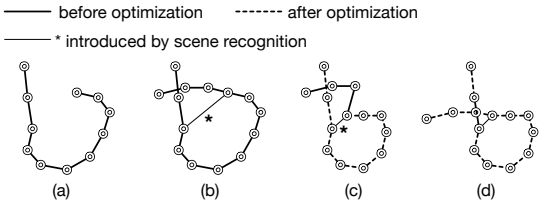

Fig. 4. (a) Local pose graph, (b) local pose graph on distributed robot is expanded, but has not yet received graph correction, (c) inconsistent graph after partial optimization, (d) corrected, globally consistent graph.

IV. EXPERIMENTS AND RESULTS

A. Relative Pose Estimation

To test how accurate the relative pose estimation is, we mounted a camera on a Kuka agilus sixx r900 industrial robotic arm with a positional accuracy of ± 0.03mm[31]. Then, we programmed the robotic arm to move along a linear rectangular path which consisted of the following motions: 20 cm to the right, 20 cm forward, 40 cm to the left, 20 cm backward, 20 cm to the right. The camera orientation was kept constant. The camera could see three of our lab walls from a distance of about 3 m. We then recorded images every 2 cm along the path. These images were used to test different relative pose calculation algorithms. Because this is a monocular setup, the scale of the relative translation was provided to the algorithms.

The results can be found in Table I and in Fig. 5. "gt" is the ground truth data, "sacN" is the 5-point algorithm by Nistér[32], "opencv" is openCV's implementation of the same 5-point algorithm, and "eightpt" is the eight-point algorithm by Longuet-Higgins[33]. The average position error is the point distance to ground truth in 3D, and the rotational error is the average of the absolute roll, pitch and yaw errors.

B. Multi-robot SLAM

We tested the overall system with two recorded datasets, named "office" and "ground floor". In these datasets, three robots are individually remote-controlled to map different parts of indoor scenery. The total mapping overlap is less than 25 m^2 each, meaning the robots mostly went separate ways. On each robot, the Karto SLAM[34] system is employed. In a few cases, single-robot SLAM shows mapping failures (Fig. 6 (a) and (b)). This is due to map registration failure of Karto. Hence, the robot must rely only on odometry for localization. This results in a weakly connected pose graph and large pose errors. Because the map registration

TABLE I

RELATIVE POSE ESTIMATION ALGORITHM RESULTS.

Algorithm	Avg. pos. err. [mm]	Avg. rot. err [deg]	Time [ms]
opencv	17.1 ± 18.1	0.20 ± 0.34	2.36
eightpt	13.2 ± 10.7	0.16 ± 0.15	0.183
sacN	11.9 ± 10.2	0.52 ± 0.86	15.9

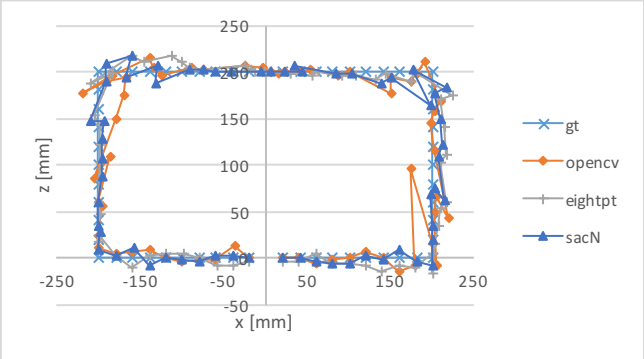

Fig. 5. Relative pose estimation illustration.

is local, the robots can not recover well from these failures without corrective data from other robots.

Figs. 1 and 8 show the individual sub-maps and their linkage through scene recognition when running the TSLAM multi-robot framework. Collaborative mapping results are seen in Figs. 7 and 9. The maps can be readily generated by overlaying the sensor range data over the pose graphs. All of these result have been generated automatically and in real-time from the recorded datasets.

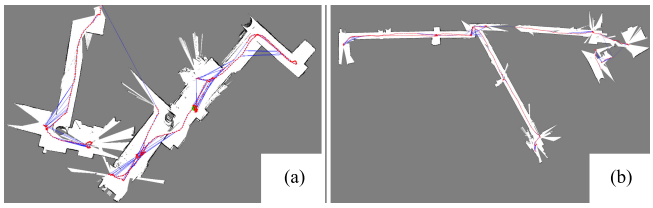

Fig. 6. Single-robot mapping failures in the (a) "office" and (b) "ground floor" dataset.

V. DISCUSSION

A. Relative Pose

We chose to use the eight point algorithm in our framework, because it offers the best balance between runtime and average error. We can use the positional error to have a rough estimate of the positional uncertainty for graph optimization.

B. Multi-Robot SLAM

We note that TSLAM was able to correctly align the data from all three robots. This means that with this framework it is possible to map the same space with three or more robots instead of one. Qualitatively speaking, this will lead to faster map coverage. It can further be seen that the mapping failures of the single-robot system could be corrected, and the robots could re-localize. This is due to TSLAM's abilities

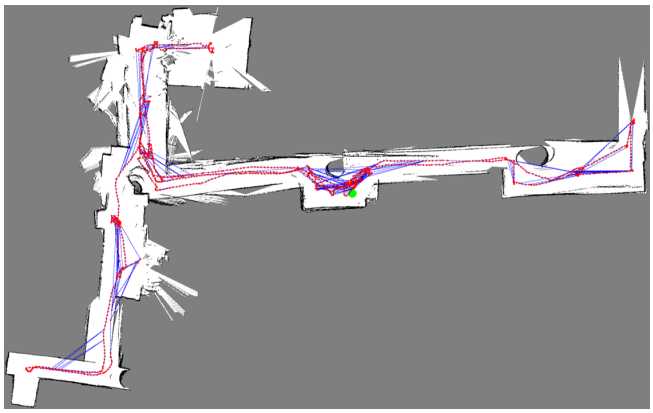

Fig. 7. The global map of the "office" dataset, merged from three robots. Total size of the map is ca. 30 m by 20 m.

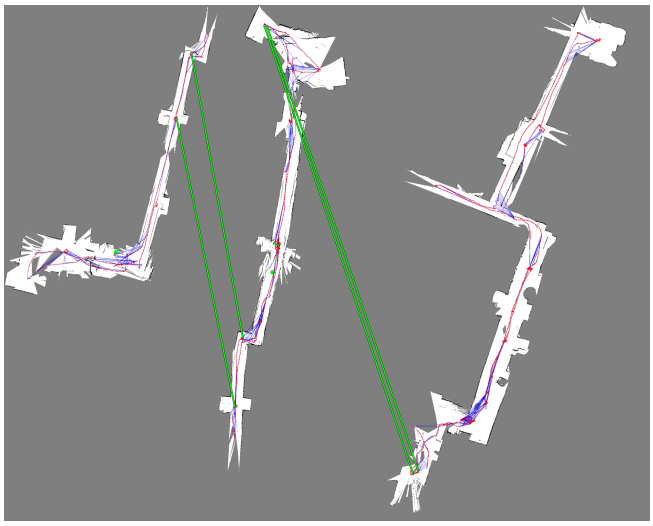

Fig. 8. The three sub-maps and their connections by scene recognition ("ground floor" dataset).

to globally correlate graphs from multiple robots, and to perform visual scene recognition (which Karto SLAM does not offer on its own). Therefore, if there is redundancy in map coverage, it can help the mapping to become more robust to failures. A shortcoming of the framework is that the combined map can only be as good as the individual sub-maps. Because Karto SLAM is not optimal in terms of map accuracy, the resulting global map looks "fuzzy" and exhibits a few angular distortions.

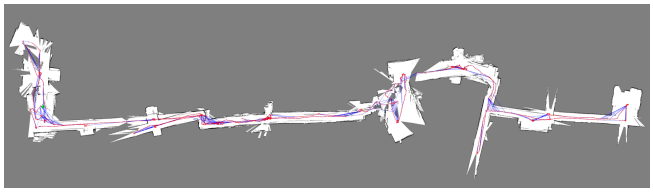

Fig. 9. The global map of the "ground floor" dataset, merged from three robots. Total size of the map is ca. 150 m by 40 m.

VI. CONCLUSION

We have presented TSLAM, a software framework that facilitates implementation of multi-robot collaborative SLAM based on existing single-robot SLAM systems. We have proposed two additions: a graph optimization consistency measure and a 3D keypoint filtering method, that complement the overall system. An application of our framework using three mobile robots has been demonstrated to have better error recovery than the equivalent single-robot case.

Future work could include taking mapping results into account for optimization, to improve the global mapping. Another topic of interest is localizing the graph optimization through topological mapping, such as done in [35], to enable large-scale application. We would also wish to conduct further experiments with mixing two SLAM systems concurrently, e.g., [7] and [5].

REFERENCES

[1] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g 2 o: A general framework for graph optimization," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 3607–3613.

[2] H. Johannsson, M. Kaess, M. Fallon, and J. J. Leonard, "Temporally scalable visual slam using a reduced pose graph," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 54–61.

[3] A. Howard, "Multi-robot mapping using manifold representations," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 4. IEEE, 2004, pp. 4198–4203.

[4] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *Intelligent Transportation Systems Magazine, IEEE*, vol. 2, no. 4, pp. 31–43, 2010.

[5] R. Mur-Artal, J. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *Robotics, IEEE Transactions on*, vol. 31, no. 5, pp. 1147–1163, 2015.

[6] 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 *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 3281–3288.

[7] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *European Conference on Computer Vision (ECCV)*, September 2014, pp. 834–849.

[8] C. Forster, M. Pizzoli, and D. Scaramuzza, "Svo: Fast semi-direct monocular visual odometry," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 15–22.

[9] R. Mur-Artal and J. D. Tardós, "Probabilistic semi-dense mapping from highly accurate feature-based monocular slam," *Proceedings of Robotics: Science and Systems, Rome, Italy*, 2015.

[10] M.-L. Doaa, A. Mohammed, M. Salem, H. Ramadan, and M. I. Roushdy, "Comparison of optimization techniques for 3d graph-based slam," *Recent Advances in Information Science*, 2013.

[11] N. Sünderhauf and P. Protzel, "Switchable constraints for robust pose graph slam," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 1879–1884.

[12] E. Olson, "Recognizing places using spectrally clustered local matches," *Robotics and Autonomous Systems*, vol. 57, no. 12, pp. 1157–1172, 2009.

[13] M. Milford, E. Vig, W. Scheirer, and D. Cox, "Towards condition-invariant, top-down visual place recognition," in *Australasian Conference on Robotics and Automation*, 2013, pp. 1–10.

[14] M. Cummins and P. Newman, "Appearance-only slam at large scale with fab-map 2.0," *The International Journal of Robotics Research*, vol. 30, no. 9, pp. 1100–1123, 2011.

[15] B. Steder, G. Grisetti, and W. Burgard, "Robust place recognition for 3d range data based on point features," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 1400–1405.

[16] C. Cadena, D. Gálvez-López, F. Ramos, J. D. Tardós, and J. Neira, "Robust place recognition with stereo cameras," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 5182–5189.

[17] A. Howard, "Multi-robot simultaneous localization and mapping using particle filters," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1243–1256, 2006.

[18] J. W. Fenwick, P. M. Newman, and J. J. Leonard, "Cooperative concurrent mapping and localization," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 2. IEEE, 2002, pp. 1810–1817.

[19] V. Indelman, E. Nelson, N. Michael, and F. Dellaert, "Multi-robot pose graph localization and data association from unknown initial relative poses via expectation maximization," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 593–600.

[20] X. S. Zhou and S. I. Roumeliotis, "Multi-robot slam with unknown initial correspondence: The robot rendezvous case," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE, 2006, pp. 1785–1792.

[21] B. Kim, M. Kaess, L. Fletcher, J. Leonard, A. Bachrach, N. Roy, and S. Teller, "Multiple relative pose graphs for robust cooperative mapping," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 3185–3192.

[22] L. Riazuelo, J. Civera, and J. Montiel, "C 2 tam: A cloud framework for cooperative tracking and mapping," *Robotics and Autonomous Systems*, vol. 62, no. 4, pp. 401–413, 2014.

[23] L. Toohey, O. Pizarro, and S. B. Williams, "Multi-vehicle localisation with additive compressed factor graphs," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 4584–4590.

[24] M. Pfingsthorn, B. Slamet, and A. Visser, "A scalable hybrid multi-robot slam method for highly detailed maps," in *RoboCup 2007: Robot Soccer World Cup XI*. Springer, 2007, pp. 457–464.

[25] A. Cunningham, K. M. Wurm, W. Burgard, and F. Dellaert, "Fully distributed scalable smoothing and mapping with robust multi-robot data association," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 1093–1100.

[26] C. Forster, S. Lynen, L. Kneip, and D. Scaramuzza, "Collaborative monocular slam with multiple micro aerial vehicles," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 3962–3970.

[27] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[28] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: an efficient alternative to sift or surf," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2564–2571.

[29] A. Glover, W. Maddern, M. Warren, S. Reid, M. Milford, and G. Wyeth, "Openfabmap: An open source toolbox for appearance-based loop closure detection," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4730–4735.

[30] L. Kneip and P. Furgale, "Opengv: A unified and generalized approach to real-time calibrated geometric vision," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1–8.

[31] Kr 6 r900 sixx (kr agilus). Accessed Feb. 14, 2016. [Online]. Available: www.kuka-robotics.com/en/products/industrial_robots/small_robots/kr6_r900_sixx/

[32] D. Nistér, "An efficient solution to the five-point relative pose problem," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 6, pp. 756–770, 2004.

[33] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms, MA Fischler and O. Firschein, eds*, pp. 61–62, 1987.

[34] Karto slam ros package. Accessed Feb. 14, 2016. [Online]. Available: wiki.ros.org/slam_karto

[35] M. Liu, F. Colas, and R. Siegwart, "Regional topological segmentation based on mutual information graphs," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 3269–3274.