Multi-Robot Localization and Mapping based on Signed Distance Functions

Philipp Koch*, Stefan May*, Michael Schmidpeter*, Markus Kühn*, Christian Pfitzner*,
Christian Merkl*, Rainer Koch*, Martin Fees*, Jon Martin*, and Andreas Nüchter†
*Faculty of Electrical Engineering, Precision Engineering, Information Technology
Nuremberg Institute of Technology Georg Simon Ohm, Germany
†Informatics VII: Robotics and Telematics
Julius-Maximilians-University Würzburg, Germany

***Abstract*—This publication describes a 2D Simultaneous Localization and Mapping approach applicable to multiple mobile robots. The presented strategy uses data of 2D LIDAR sensors to build a dynamic representation based on Signed Distance Functions. A multi-threaded software architecture performs registration and data integration in parallel allowing for drift-reduced pose estimation of multiple robots. Experiments are provided demonstrating the application with single and multiple robot mapping using simulated data, public accessible recorded data as well as two actual robots operating in a comparably large area.**

I. Introduction

Rescue forces risk their own health and life in service for people in serious trouble. In collapsed or burning buildings the search for victims is dangerous and time critical. In general, such disaster sites can be assumed as unknown areas wherefore the search for injured or trapped persons is an exploration task in the first place. Included autonomous robots face the well-known Simultaneous Localization and Mapping (SLAM) problem.

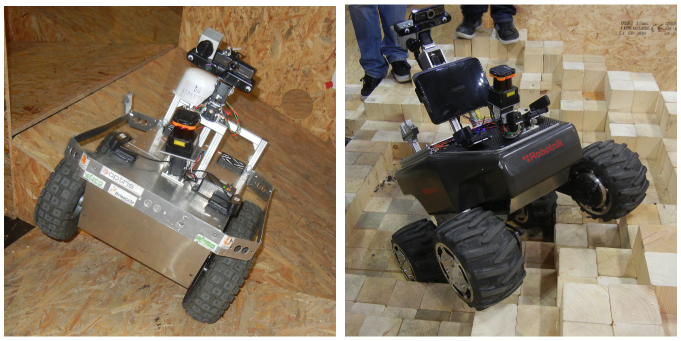


Fig. 1: Cooperating Robots at the RoboCup German Open 2014.

The chance to save lives reduces gradually in time. Efficient search is performed in parallel. Multiple rescue forces enter the disaster area from different access points. In order to support human rescue forces in dangerous tasks with multiple robots, coordination and data fusion is mandatory to defragment collected sensor data of several robots. Piecing together these fragments is required for getting a general idea about the whole situation. Rescue forces are instructed more efficiently having the global overview.

In this paper, we propose a 2D multi-SLAM framework allowing a robot team to cooperate together for drift-reduced pose estimation and shared mapping. A multi-threaded software architecture allows parallel pose estimation of multiple 2D LIDAR inputs. Either a shared map or individual maps can be used for all robot instances.

One could mention that a 2D algorithm is not sufficient regarding the comparably structured area of a search and rescue site. The proposed approach is derived from our previously published framework which allows application to 2D and 3D. However, as this publication focuses on multi robot SLAM of large areas which would require a high amount of system resources, 2D perception is performed. Nevertheless, a 2D map is easier to read for human rescue personal in high stress situations.

The content of this paper is structured as follows: Section II reviews related work in multi-robot SLAM. Section III outlines our framework and develops the concrete model for including 2D laser scanners. Section IV extends the framework for application to multi-robot SLAM. In Section V single-robot and multi-robot SLAM experiments are performed, either in simulation and in a real world scenario. Finally, Section VI concludes with an outlook on future work.

II. Related Work

In an early approach, Burgard et al. considered multiple robots as independent systems to be coordinated for a faster coverage of the exploration area [1]. The global map is a result of integrating several local maps. If their relative poses are known, map integration is straightforward. While focusing on the collaboration aspect, an extended approach still makes need of close initial robot poses [2].

Several probabilistic models were proposed to solve the problem of unknown starting poses of multiple robots in a joined exploration task. Konolige et al. used local probabilistic constraints among robot poses [3]. Each pair of the robot team exchanges local maps for merging both to obtain a globally consistent map with loop-closure techniques. Howard proposed Rao-Blackwellized particle filters for the localization problem [4]. Local maps are merged as the robots bump into each other during mission.

Fox et al. demonstrated the efficient exploration of unknown environments by a team of mobile robots [5]. Also this approach does not rely on initial pose information about the robot team members. Having the ability to explore independently, the robots can build clusters in order to share a map as soon as they come close enough for establishing stable communication links. The robots need to determine their pose relative to the coordinate system of the shared map. Particle filters are employed for this task on each robot. The proposed approach works efficiently for up to six robots. Global consistency is ensured by the application of loop-closure techniques.

The approach of Kim et al. relies on multiple relative pose graphs for the cooperative mapping task with a team of mobile robots [6]. Real-time applicability was achieved as well as fast convergence to a global solution. The approach also employs loop-closure detection. Kim et al. illustrated the performance in larger environments, of which several thousand laser scans were provided. Different kinematic concepts were included, i.e., the cooperative mapping using a quadrotor and a ground robot.

Granström et al. proposed the detection of loop closures with a machine learning approach [7]. The group used AdaBoost for building rotation invariant features detected in laser scans.

In 2D SLAM approaches loop-closure detection plays an important role. Interestingly, the 3D SLAM approach KinectFusion has no need to rely on this step. Izadi et al. demonstrated that the representation of a Signed Distance Function (SDF) [8] applied to the SLAM problem, achieves accurate tracking results with limited drift [9]. The group achieved real-time capability by the use of massive parallelism on GPU. A hand-held Kinect was localized by Iterative Closest Point (ICP) registration while tracking against the growing full surface model. The convergence of the system without explicit global optimization was demonstrated in several closed-loop scenarios. This approach aims at the assimilation of as many surface measurements as possible in time and features a limited drift and high accuracy.

In a previous publication, we generalized the KinectFusion approach in order to make it applicable to different types of sensors, i.e., 2D and 3D laser scanners, Time-of-Flight and stereo cameras or structured light sensors [10]. In this publication we extend this framework for the SLAM problem performed by multiple robots. The approach can either be applied to several robots independently or to a team of robots sharing and updating a joined map.

III. Algorithm

For the reader's convenience we explain in this Section the application of the generic framework to 2D laser scanners [10]. The multi-robot extension and closed-loop experiments follow subsequently.

An iteration of our *SLAM* approach consists of three steps, it is triggered by new sensor frames. In the first step, the physical parameters of the input device are used to reconstruct a model $\mathbf{M} = \{\vec{m}_i \mid i = 1..n_m\}$ containing coordinates $\vec{m}_i = (x_i, y_i)^T$, a virtual sensor frame from the pose the sensor was previously localized at.

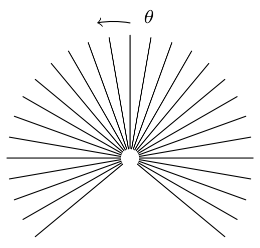


Fig. 2: Raycasting model for 2D laser range finder.

Step two uses this data as model for scan matching with the current sensor data, the scene $\mathbf{D} = \{\vec{d}_i \mid i = 1..n_d\}$ containing coordinates $\vec{d}_i = (x_i, y_i)^T$, deploying the ICP algorithm introduced by Zhang and Zhengyou [11] and Chen and Medioni [12]. The sensor's pose, denoted as 3×3 transformation matrix $\mathbf{T}_i$, is updated with incremental pose change $\mathbf{T}^*$ from time step $i-1$ to i.

The third step uses the current pose and sensor data to update the representation by calling the method `push`, see Figure 3. The grid contains Truncated Signed Distances (TSD) similar to the KinectFusion approach [9]. We call this representation TSD grid in the remainder. Listing 1 illustrates coarsely the whole approach.

Algorithm 1 Main *SLAM* Strategy

```
$\mathbf{D} \leftarrow$ acquire sensor data
$\mathbf{M} \leftarrow$ reconstruct: raycast at pose $\mathbf{T}_{i-1}$
$\mathbf{T}^* \leftarrow$ scan registration between $\mathbf{D}$ and $\mathbf{M}$
$\mathbf{T}_i \leftarrow \mathbf{T}^*\mathbf{T}_{i-1}$
Integrate data $\mathbf{D}$ at pose $\mathbf{T}_i$
```

Localization of multiple robots can be performed in parallel, since raycasting and scan matching apply only reading access. The *SLAM* strategy of a single robot can be parallelized itself, because map updating is executed only in small incremental steps. The possible error induced by a conflicting read and write access can be neglected, as we assimilate as many surface measurements as possible in time, cf. KinectFusion approach [9].

A. Reconstruction

A representation based on SDF has the characteristic, that raycasting can be employed to reconstruct scans from arbitrary point of views. Data integrated from multiple views are weighted with a decreasing weight. Thus, the reconstruction from the TSD grid at a certain point of view entails information of all integrated scans so far and features reduced noise. The sensor model for the raycaster defines a set of vectors, i.e., the line of sight of each laser beam, cf. Fig. 2.

B. Data Integration

The SDF is calculated for every grid cell visible by the sensor, wherefore the first step is back projecting the cell centroids $\mathbf{V} = \{\vec{v}_i \mid i = 1..n_v\}$, i.e., assigning a certain laser beam. As these coordinates are in the world coordinate system, they need to be registered to the sensor coordinate system as follows:

$$\mathbf{V}^* = \mathbf{T}_i^{-1}\mathbf{V} \tag{1}$$

The centroids $\vec{v}_i^*$ are assigned to laser beams as follows:

$$\alpha_i = \arctan(\frac{v_{iy}^*}{v_{ix}^*}), \tag{2}$$

$$i_i = \frac{\alpha_i}{r} \tag{3}$$

where α_i is the beam's polar angle, i_i the assigned beam index and r the sensor's angular resolution.

IV. Multi-Robot Framework

As described in the previous Section, an iteration of the *SLAM* approach consists mainly of three steps: reconstruction, localization and data integration. Considering the usage of a Robot Operating System (ROS)-based architecture, a fourth step is necessary extracting a compatible representation out of the TSD grid. Many ROS nodes require an occupancy grid as input.

Considering simultaneous multi-SLAM capabilities, these tasks have to be performed for every robot. However, as the current pose needs to be supplied with a fast and constant update rate in order to use it for pose and motion controllers, the localization should be decoupled from other tasks. Modern CPUs consist in general of multiple cores allowing parallel processing of data with a multi-threaded architecture.

To supply a fast pose update rate, a high priority localization thread is started for each robot, which is triggered by new input data. Map building (data integration) is executed asynchronously because it only needs to be performed if the pose changes significantly. Most navigation tools in ROS do not require a high update rate for the map wherefore the occupancy grid generation is triggered by a timer at a comparably low rate.

The framework provides three thread classes: a localization thread (`ThreadLocalize`) is instantiated for every robot. Grid extraction (`ThreadGrid`) and data integration (`ThreadMapping`) is performed each by a single thread as well. The data integration module provides a queue to relax heavy work load, when multiple robots desire to integrate their data to the shared map. Fig. 3 depicts the coarse framework.

A. Localization Thread

The model for ICP matching is reconstructed from the TSD grid, which is used simultaneously by all robots. As only reading access is performed, no racing conditions can appear, even if multiple accesses to the same cell are performed. Therefore, the multi-robot-SLAM approach uses one separate localization thread for each robot.

B. Mapping Thread

All localization threads access the same instance of the representation wherefore these accesses have to be synchronized. Our framework uses an additional thread for this task, which updates the representation with a given data set.

As the localization cycle time is the most crucial, these threads must not wait for the map update. Therefore, our approach uses a First in First Out buffer to which access is controlled by a mutex. Multiple localization threads add data to the queue wherefore waiting is restricted to the copying of the data. Typical 2D laser frames contain approximately 1000 points, wherefore the time for copying is negligible.

An instance of our sensor interface includes this information. Sensor data is handled using instances of this interface. The mapping thread remains inactive until data is added to the queue. Its event loop updates the map with the sensor data until the queue is empty.

However, as multiple robots add data to the map, it has to be ensured that an active robot is not mapped as an obstacle by another robot. For instance, consider one robot following another. Therefore, a filter takes the positions and footprints of all robots into account. This filter applies two steps:

First, the corners of their footprints are back projected to assign laser beams of the sensor's model (Equations (1)-(3)). Gained indices are verified whether they are in the sensor's field of view, otherwise the referring robot is not visible and not to be considered any further in this iteration.

Second, the distance of the footprint corners to the referring measurements are compared. This is necessary in order to detect occlusion by another object. If the footprint corners are visible, the associated depth data is not taken into account in the map building process. This approach is illustrated in Algorithm 2.

Algorithm 2 Robot pose filter

```
for all known robot positions with robot radius $r_i$ do
    for all robot footprint corners $\mathbf{C} = \{\vec{c}_i \mid i = 1..n_c\}$ do
        $\vec{c}_i^* \leftarrow \mathbf{T}_i^{-1}\vec{c}_i$
        $i_i \leftarrow \frac{\arctan(\frac{c_{iy}}{c_{ix}})}{r_i}$
        if ($i_i$ within sensor bounds) then
            if ($D(i_i) > \left\|\vec{c}_i - \vec{t}_i\right\|$) then
                exclude $D(i_i)$
            end if
        end if
    end for
end for
```

C. Occupancygrid Thread

In order to supply map data represented as occupancy grid, the TSD grid content has to be translated into occupancy likelihoods, wherefore an additional thread is integrated performing this task.

The SLAM approach knows three possible cell states: unknown, empty or occupied, i.e., containing an object. The ROS definition of an occupancy grid sets unknown areas to -1 and empty areas to 0. A cell containing an object with $100\,\%$ likelihood is marked with the value 100.

As surfaces are determined by a sign change in the SDF, reconstruction is performed by raycasting. Our approach applies axis-aligned rays traversing from two adjacent sides of the grid to their opposite side. On the contrary to the reconstruction

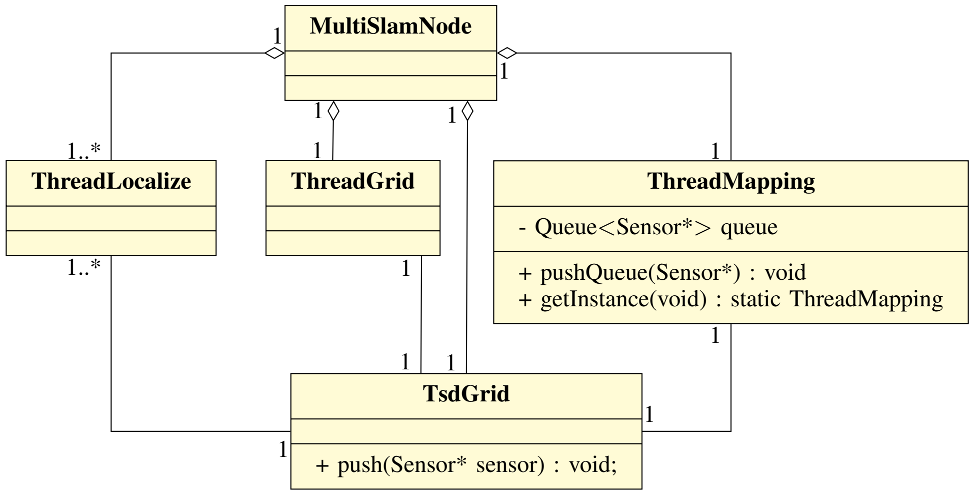

MultiSlamNode
ThreadLocalize
ThreadGrid
ThreadMapping
- Queue<Sensor*> queue
+ pushQueue(Sensor*) : void
+ getInstance(void) : static ThreadMapping
TsdGrid
+ push(Sensor* sensor) : void;
1
1..*

Fig. 3: UML diagram of the thread architecture.

from a certain point of view, this approach does not aim at getting a virtual sensor frame, it reconstructs the complete content of the representation at once.

V. Experiments

All experiments described in the following were performed on the same type of CPU, an Intel Core i7 quad core. As operating system, Ubuntu 14.04 lts with ROS Indigo was chosen.

A. Single Robot Loop Closure Experiment

In this experiment our rescue robot "Simon" (Figure 9 a), equipped with a Hokuyo UTM-30LX LIDAR, was navigated through the first floor of a building at our campus. The test was aggravated through small floors consisting of walls with few distinctive features and a noticeable amount of uncovered glass surfaces. Additionally, start- and end point are closely located. Loop closures could be detected, if applied. This is done implicitly as the TSD grid weights all incoming data appropriately and features minimal drift. The map has an edge length of 122.88 m and a granularity of 0.015 m.

Figure 4 shows four steps of the loop closing, illustrating a comparably small error as the robot arrives at its starting point again. Figure 5 shows the final map with drawn trajectory of this experiment.

B. Single Robot SLAM with Reference Data

A second experiment deploying a single robot has been performed in order to test our SLAM framework. As input data, reference laser frames from a data repository at the University of Freiburg, Germany, were used [13]. In order to validate the output of the software, an image of the resulting map with printed trajectory is provided, cf. Figure 6a, as well as the reference map and trajectory supplied by the University of Freiburg, cf. Figure 6b.

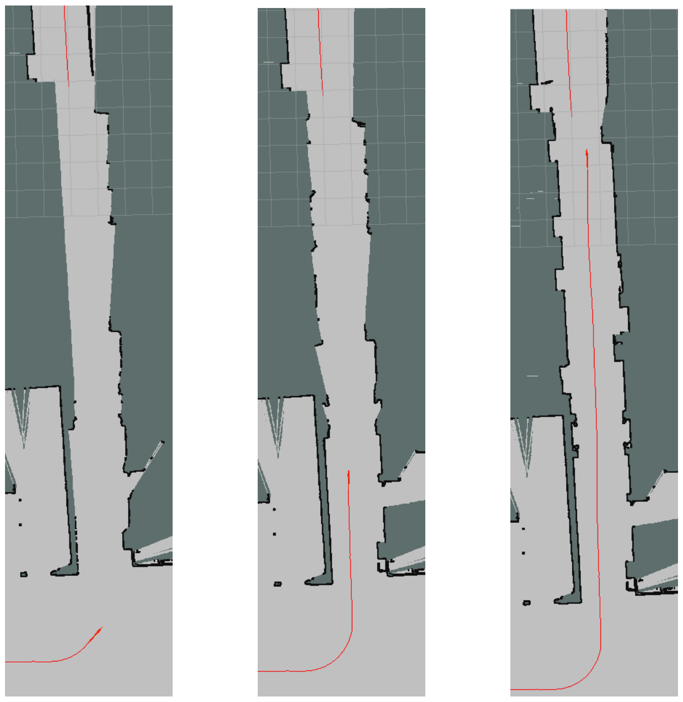


Fig. 4: Three steps of the loop closing, chronological order from left to right.

C. Multi SLAM with Simulated Robots

The multi-SLAM framework has been tested with the ROS Simple Two Dimensional Robot Simulator (STDR)[1] . The simulator is installed on an additional PC, it provides artificial laser data for every robot. As the main processor is a quad core, the number of robots for this experiment has been set to the number of physical processor cores, wherefore hardware resources are used at high capacity. The map has an edge length of 122.88 m and a granularity of 0.015 m.

[1]http://wiki.ros.org/stdr_simulator, online accessed 14-January-2015

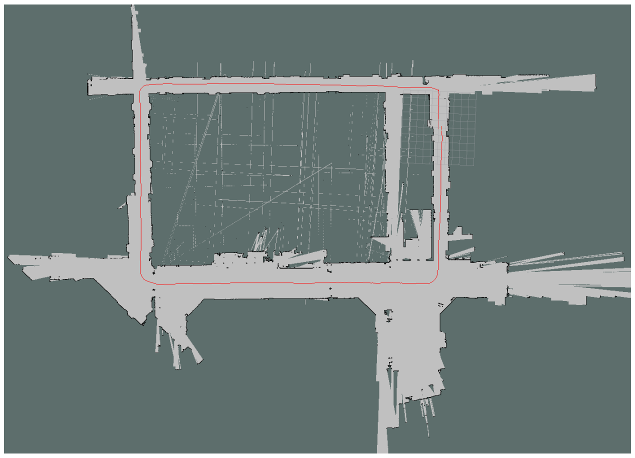
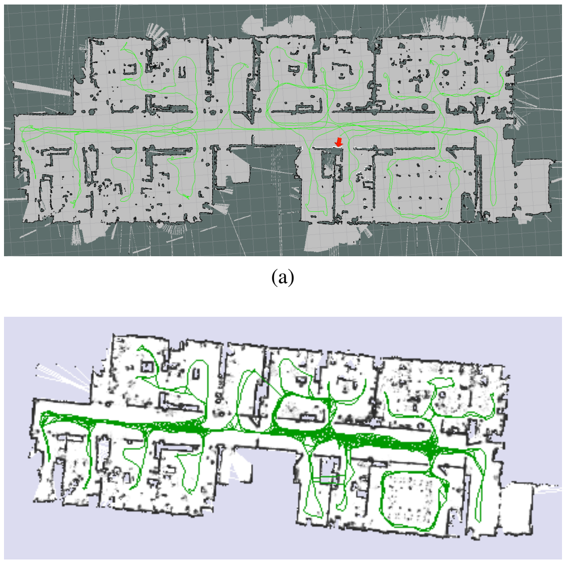


Fig. 5: Complete map of the single slam loop closure experiment with drawn trajectory.



(a)

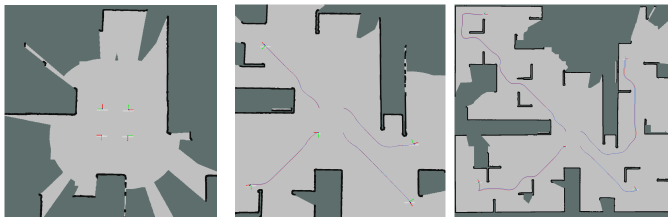
(b) *source:* http://kaspar.informatik.uni-freiburg.de/ slamEvaluation/datasets.php

Fig. 6: Result of the reference data experiment. Fig. (a) represents the generated map with estimated trajectory, (b) shows a reference image taken from the data repository.

The four simulated units start at the same time and explore a labyrinth, building a map of the surrounding. The experiment was documented taking screenshots of the map containing the ground truth of the simulator and the estimated trajectory. Figure 7 and 8 show the process and the results of the simulated experiment. The blue line marks the ground truth, and the red line the estimated trajectory.

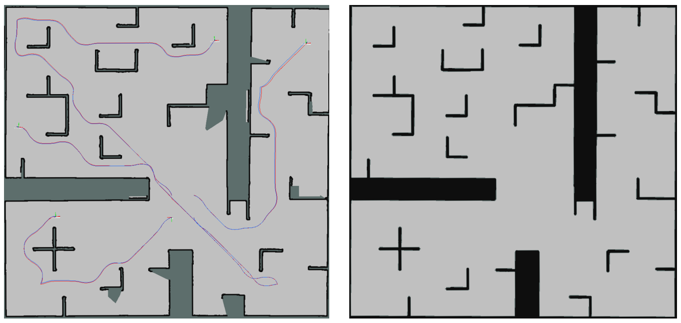
D. Multi SLAM of a Building Floor

This experiment addresses the RoboCup Rescue scenario, the multi-SLAM is being developed for a team of two cooperating robots (Figure 9) exploring an indoor area. The robot "Simon" explores one part of the building, and robot "Georg" another. Both are equipped with the same LIDAR, a Hokuyo UTM30-LX. In order to validate the limited drifting error of our framework, both trajectories contain loops. The mapped building is the same as in section V-A.



(a) (b) (c)

Fig. 7: Phases of the simulated multi-SLAM. Chronological order from a to c.



(a) (b)

Fig. 8: Multi-SLAM simulation result. Comparison of the reconstructed map (a) and the original model used by the simulator (b).

(a) (b)

Fig. 9: Multi-SLAM with two cooperating robots. Image showing "Simon" (a) and "Georg" (b) during the multi-SLAM experiment.

Figure 10 illustrates both robots closing their loops simultaneously as they arrive at the same time at their starting points. These images depict the limited drift of our framework as only comparably small errors occur.

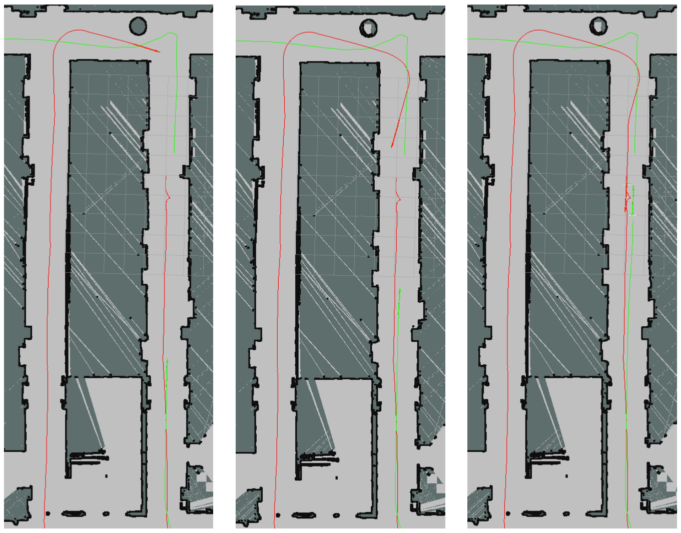


Fig. 10: Multi-SLAM loop closure. Image illustrating three steps of the robots closing their loops simultaneously.

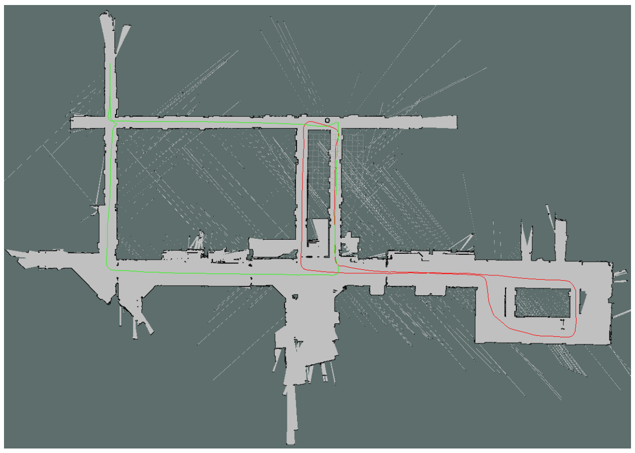


Fig. 11: Complete map of the cooperative mapping. Red color marks Simon's trajectory, green Georg's trajectory.

VI. Conclusion and Future Work

In this publication, we presented our multi-SLAM framework. We illustrated how our previous work [10] is extended to a simultaneous multi source localization and mapping application.

The paper provides experiments which depict the localization with limited drift in single- and multi-SLAM mode, with simulated and real data. Future work will include an accuracy analysis with an exact ground truth in order to evaluate the estimated trajectory. Additionally, a detailed timing evaluation of the approach is required. As the approach already demonstrated its capabilities with our rescue robot team, it will be deployed at the RoboCup Rescue German Open 2015 competition.

The software is open source and available at http://www.github.com/autonohm/obviously.git and http://www.github.com/autonohm/ohm_tsd_slam.git.

References

[1] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun. Collaborative multi-robot exploration. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 1, pages 476–481 vol.1, 2000.

[2] W. Burgard, M. Moors, C. Stachniss, and F.E. Schneider. Coordinated multi-robot exploration. *Robotics, IEEE Transactions on*, 21(3):376–386, June 2005.

[3] Kurt Konolige, Dieter Fox, Charlie Ortiz, Andrew Agno, Michael Eriksen, Benson Limketkai, Jonathan Ko, Benoit Morisset, Dirk Schulz, Benjamin Stewart, and Rgis Vincent. Centibots: Very large scale distributed robotic teams. In Marcelo H. Ang Jr. and Oussama Khatib, editors, *ISER*, volume 21 of *Springer Tracts in Advanced Robotics*, pages 131–140. Springer, 2004.

[4] Andrew Howard. Multi-robot simultaneous localization and mapping using particle filters. In *Proceedings of Robotics: Science and Systems*, Cambridge, USA, June 2005.

[5] Dieter Fox, Jonathan Ko, Kurt Konolige, Benson Limketkai, Dirk Schulz, and Benjamin Stewart. Distributed multi-robot exploration and mapping. In *Proceedings of the IEEE*, page 2006, 2006.

[6] Been Kim, Michael Kaess, Luke Fletcher, John Leonard, Abraham Bachrach, Nicholas Roy, and Seth Teller. Multiple relative pose graphs for robust cooperative mapping. In *IEEE Intl. Conf. on Robotics and Automation, ICRA*, pages 3185–3192, May 2010.

[7] K. Granstrom, J. Callmer, F. Ramos, and J. Nieto. Learning to detect loop closure from range data. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 15–22, May 2009.

[8] Stanley Osher and Ronald Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces (Applied Mathematical Sciences)*. Springer, 2003 edition, November 2002.

[9] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. KinectFusion: Real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, 2011.

[10] Stefan May, Philipp Koch, Rainer Koch, Christian Merkl, Christian Pfitzner, and Andreas Nüchter. A generalized 2d and 3d multi-sensor data integration approach based on signed distance functions for multi-modal robotic mapping. In *VMV 2014: Vision, Modeling & Visualization, Darmstadt, Germany, 2014. Proceedings*, pages 95–102, 2014.

[11] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *Int. J. Comput. Vision*, 13(2):119–152, October 1994.

[12] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 2724–2729 vol.3, Apr 1991.

[13] slam benchmarking. http://kaspar.informatik.uni-freiburg.de/~slamEvaluation. Online; accessed 14-January-2015.