J Intell Robot Syst
DOI 10.1007/s10846-016-0375-7

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 · Daniel Ammon · Andreas Nüchter

Received: 7 June 2015 / Accepted: 1 May 2016

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. Novelties of the approach are a joint map built in parallel instead of occasional merging of smaller maps and the limited drift localization which requires no loop closure detection. 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, two actual robots operating in a comparably large area as well as a deployment of these units at the Robocup rescue league.

Keywords Mobile robotics · Multi-robot · Rescue robotics · SLAM

P. Koch (✉) · S. May · M. Schmidpeter · M. Kühn · C. Pfitzner · C. Merkl · R. Koch · M. Fees · J. Martin · D. Ammon
Faculty of Electrical Engineering, Precision Engineering, Information Technology, Nuremberg Institute of Technology Georg Simon Ohm, Nuremberg, Germany
e-mail: philipp.koch@th-nuernberg.de

A. Nüchter
Informatics VII: Robotics and Telematics, University Wuerzburg, Würzburg, Germany

1 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.

The chance to save lives reduces gradually in time, wherefore multiple rescue forces searching in parallel increase the efficiency. In order to support human teams 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.

The invention of robots capable of deployment in rescue scenarios is a complicated and therefore expensive task. Inspired by catastrophes resulting in huge amounts of injured or buried people, the Robocup rescue league was founded.

This organisation aims at turning rescue robotics into a world wide open source project. In general, teams of students working at universities or institutes of technology create robots which participate at national and international competitions. After these events, the teams publish their results and allow public access to their software, mechanical design and other technical details.

This leads to a world wide open source community as all teams can base further research on these results. Examples for robots competing in this league are illustrated in Fig. 1.

In order to simulate a disaster scenario at the Robocup rescue competition, a maze is used consisting of standardized components such as ramps, stairs or random step fields. In the future, 3D perception will be necessary in order to create a system capable of prevailing in a real scenario but currently 2D localization and mapping is sufficient for the Robocup rescue league.

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 in the Robocup rescue competition. A main novelty compared to other approaches is the joint map built in parallel by the cooperating robots.

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. Nevertheless the proposed approach aims at deployment in the Robocup rescue league for which 2D perception is currently sufficient.

The content of this paper is structured as follows: Section 2 reviews related work in multi-robot SLAM. Section 3 outlines the proposed framework and develops the concrete model for including 2D laser scanners. Section 4 extends the framework for application to multi-robot SLAM. In Section 5 single-robot and multi-robot SLAM experiments are performed, either in simulation and in a real world scenario. In Section 6, the application of the approach at the Robocup German Open 2015 is described. Finally, Section 7 concludes with an outlook on future work.

This paper is an extension of already published work [11]. The extensions consist of a ground truth analysis concerning public available reference data, as well as simulated data. The multi-SLAM has been deployed at the Robocup German Open, wherefore the paper contains data of the competition as well as lessons learned. The framework has been compared to the current state of the art 2D-SLAM approach at the Robocup, using similar parameters and the same input data. Additionally, the multi-SLAM experiment deplyoing a team of two robots has been enhanced with a timing analysis of the framework.

2 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 [3]. 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 [4].

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 [13]. Each pair of the robot team exchanges local

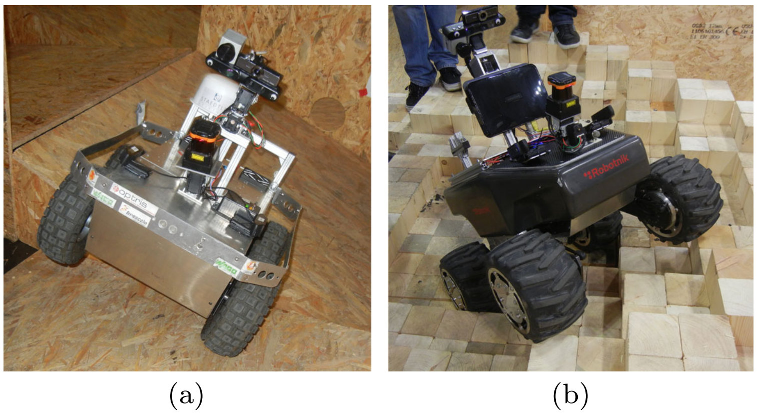


Fig. 1 Cooperating Robots at the Robocup German Open 2014

maps for merging both to obtain a globally consistent map with loop-closure techniques. Howard proposed Rao-Blackwellized particle filters for the localization problem [8]. Local maps are merged in case robots come close to each other during mission.

Fox et al. demonstrated the efficient exploration of unknown environments by a team of mobile robots [6]. 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 [10]. 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, e.g., 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) [16] 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, e.g., 2D and 3D laser scanners, Time-of-Flight and stereo cameras or structured light sensors [15]. 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.

3 Algorithm

For the reader's convenience, this section contains application of the generic framework to 2D laser scanners [15]. The multi-robot extension and closed-loop experiments follow subsequently.

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

Step two uses this data as model for scan matching with the actual laser measurements, the scene $\mathbf{D} = \{\mathbf{d}_i \mid i = 1..n_d\}$ containing coordinates $\mathbf{d}_i = (x_i, y_i)^T$. We use the well known Iterative Closest Points (ICP) algorithm introduced by Zhang and Zhengyou [17] and Chen and Medioni [5]. The sensor's pose is denoted as 3×3 transformation matrix $\mathbf{T}_i$, consisting of a translational vector $\mathbf{t} = (t_x, t_y)^T$ and a rotational matrix $R(\alpha)$. $\mathbf{T}_i$ is updated with incremental pose change $\mathbf{T}^*$ from time step $i-1$ to i:

$$\mathbf{T}_i = \mathbf{T}^*\mathbf{T}_{i-1}, \mathbf{T}^* = \begin{pmatrix} cos(\alpha) & sin(\alpha) & t_x \\ -sin(\alpha) & cos(\alpha) & t_y \\ 0 & 0 & 1 \end{pmatrix}.$$

The third step uses the current pose and sensor data to update the representation. The grid contains Truncated Signed Distances (TSD) as used in the well known KinectFusion approach [9]. We call this representation TSD grid in the remainder. Algorithm 1 illustrates coarsely the whole approach.

Pose estimation of multiple robots has to be executed in parallel. Since the sensor data is matched against the global map, parallel access is inevitable. Nevertheless, raycasting reconstruction applies only reading access wherefore synchronization using mutexes is unnecessary.

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 $\mathbf{T}_i$
```

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].

3.1 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. Figure 2a.

In case of a 2D LIDAR, all raycaster peak points $\mathbf{P} = \{\mathbf{p}_i \mid i = 1..n_p\}$ are located on the unit circle wherefore their calculation is straight forward using trigonometric functions. In essence, the polar data of the laser scanner is converted into Cartesian vectors.

The directional vector $\mathbf{v}_i = \mathbf{p}_i - \mathbf{t}_i$ and the origin, the current translation $\mathbf{t}_i$, are registered into the world coordinate system applying the current transformation $\mathbf{T}_i$. The raycaster is sent in steps through the representation, the step width is achieved by normalizing v_i to the desired length and adding this value to the starting pose in every iteration.

This method iterates until the raycaster either reaches a border layer of the representation or detects a certain sign change in the SDF. The described approach uses positive signs in front of an object and negative sings behind it. Therefore, only a sign change from positive to negative refers to an object. An irregular sign change (from negative to positive) is regarded as an artefact in the representation and the return value of the raycaster is neglected in this iteration.

As objects are represented by two layers of cells with different signs, it is not likely that the raycaster skips the object. This is a major upside of an SDF based representation compared to a Cartesian cell based approach. The computed coordinates contain a systematic error as the sign change can be detected several cell layers behind the actual object, wherefore an interpolation is applied considering the SDF in the neighbouring cells. Algorithm 2 summarizes the described process.

3.2 Data Integration

The SDF is calculated using Eq. 1 which is illustrated in Fig. 2b as well. The distance between the centroid of the current cell $\mathbf{v}_i$ and the referring distance measurement $D(i_i)$ is signed positive in front of an object

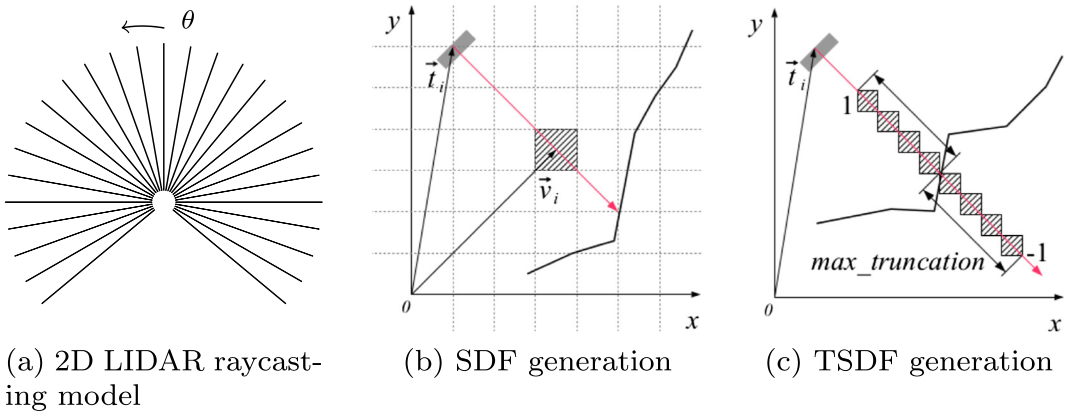


Fig. 2 Reconstruction and Mapping. **a** Illustrates the raycasting model for a 2D LIDAR. **b** Depicts the generation of the SDF. The current translation of the sensor is marked by $\mathbf{t}_i$. **c** Shows the truncation to TSDF

Algorithm 2 Raycasting based reconstruction

```
Calculate peak points $\mathbf{p}_i$
Calculate $\mathbf{v}_i$
Normalize $\mathbf{v}_i$ to step length
$\mathbf{t}_i^* = \mathbf{T}_i \mathbf{t}_i$
while (raycaster within bounds) do
    if ($sdf_{i-1} > 0$ & $sdf_i < 0$) then
        Interpolate coordinates
        Return
    end if
end while
```

or negative, behind it. This computation is depicted in Fig. 2b.

$$sdf_i = \|\mathbf{t}_i - \mathbf{v}_i\| - D(i_i) \tag{1}$$

In order to determine the referring laser beam, the cell centroids $\mathbf{V} = \{\mathbf{v}_i \mid i = 1..n_v\}$ are back projected with a specific sensor model. This computation calculates the laser beam closest to the current cell centroid $\mathbf{v}_i$, it is depicted in Eq. 3.

As these coordinates are in the world coordinate system, they need to be registered to the sensor coordinate system as follows:

$$\mathbf{V}^* = \mathbf{T}_i^{-1}\mathbf{V} \tag{2}$$

The centroids $\mathbf{v}_i^* = (v_{ix}^*, v_{iy}^*)$ are assigned to laser beams as follows:

$$\alpha_i = \arctan(\frac{v_{iy}^*}{v_{ix}^*}), \tag{3}$$

$$i_i = \frac{\alpha_i}{r} \tag{4}$$

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

In order to determine whether the back projected cell centroid is visible by the sensor from the current pose, the computed beam index is compared to the sensor bounds. These bounds refer to the maximal and minimal beam indices and are defined by the physical parameters of the sensor model. If this index complies the comparison, the current SDF is computed, otherwise the algorithm continues with the next cell centroid.

In order to prevent occlusions of objects, the SDF is truncated to a certain cell layer thickness, the so called truncation radius. This process is illustrated in Fig. 2c. The output is stored in the cell using a weighted average. Algorithm 3 illustrates the approach.

Algorithm 3 Map update

```
$\mathbf{V}^* = \mathbf{T}_i^{-1}\mathbf{V}$
Back project $\mathbf{V}^*$
for (all $v_i^*\mathbf{V}^*$) do
    Calculate beam index $i_i$
    if ($i_i$ within bounds) then
        $sdf_i = \|\mathbf{t}_i - \mathbf{v}_i^*\| - D(i_i)$
        Calculate weighted average
        Store average and new weight
    end if
end for
```

4 Multi-Robot Framework

As the multi-SLAM framework uses a TSD based representation, it will be referred to as TSD SLAM in the following. As described in the previous section, an iteration of the TSD 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. The described strategy does not merge the generated maps of the cooperating robots, it uses the ability of the framework, to update the map from a certain pose. Therefore, every cooperating robot is able to add new sensor data to the global shared map.

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, for instance 0.5 Hz.

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. Figure 3 depicts the coarse framework.

4.1 Localization Thread

The model data set 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 at the same time. Therefore, the multi-robot-SLAM approach uses one separate localization thread for each robot.

Theoretically, robots are able to start from every pose within the bounds of the map, as long as this initial pose is known. Therefore, robots could enter the area from different edges of the map and explore without ever getting close to each other. However, as the proposed approach aims at deployment at the Robocup rescue, cooperating robots enter the area at similar positions.

The initial laser scan of the first starting robot is used to initialize the map from the initial pose of the robot. The data of any other starting mobile unit needs to be localized in the map, using its set initial pose as a pre-transformation for the scan matching.

4.2 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 most crucial, these threads must not wait for the map update. Therefore, our approach uses a First in First Out (FIFO) 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.

A concrete instance of our sensor interface is used to handle the information. 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. The mapping thread uses the sensor data and the current localization of each cooperating robot to update the map sequentially from the given pose.

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. However, such a moving obstacle is hard to distinguish from static objects but as long as both robots operate on the same multi-SLAM server, their poses are known. Therefore, the most external points of their frames, in the following described as corners, are used by a filter to mask

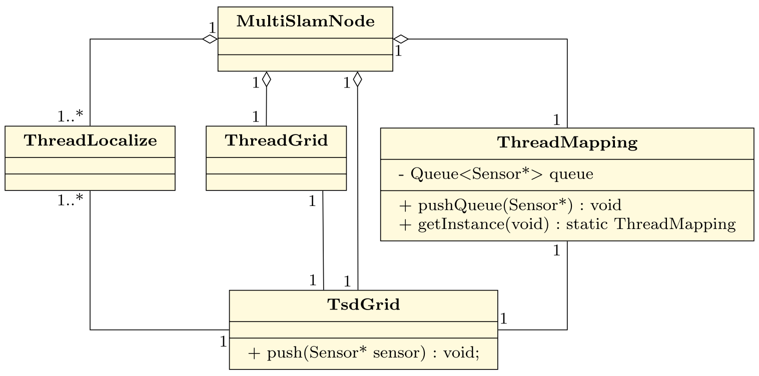

MultiSlamNode
ThreadLocalize
ThreadGrid
ThreadMapping
- Queue<Sensor*> queue
+ pushQueue(Sensor*) : void
+ getInstance(void) : static ThreadMapping
TsdGrid
+ push(Sensor* sensor) : void;
1
1..*
1

Fig. 3 UML diagram of the threaded architecture

the laser beams clipping the area restricted by these corners. This module takes the positions and footprints of all robots into account, it applies two steps:

First, the corners of their footprints are back projected to assign laser beams of the sensor's model (2–4). 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 4.

Algorithm 4 Robot pose filter

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

4.3 Occupancy Grid 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 sign changes in the SDF, a raycasting approach is used to reconstruct the map content. On the contrary to the already described local raycasting in chapter 3.1, a global strategy is required.

Our approach applies axis-aligned rays traversing from two adjacent sides of the grid to their opposite side. Rays are started in the origin of every cell of the restricting layers, running parallel to the x- or y-axis but do not terminate after detecting an object. Each sign change is used to interpolate a point in the map coordinate system, resulting in a 2D point cloud representing the complete map content.

This data is used to fill in 100 % likelihoods into the occupancy grid by discretizing the coordinates with the occupancy grid resolution. However, this strategy lacks the definition of empty or unknown areas, wherefore the axis parallel running raycaster has to add additional information.

As these rays already have to examine the SDF value in all cells of the representation, this information can be used to add the necessary data. Our representation consists of partitions which are only initialized when data is added, wherefore every uninitialized partition leads to an unknown area in the occupancy grid. During the initialization of a partition, every yet unknown cell contains a `Not a Number (NaN)` value, which is read by the raycaster and is also translated into an unknown cell in the occupancy grid.

All cells which contain a valid SDF value are marked as empty. As sign changes in this values mark objects, this assumption is not completely accurate. Nevertheless, these cells are overwritten with the points generated by the raycasters, which corrects this error.

4.4 Communication

The communication between all robots connected to the multi-SLAM system is provided by the ROS framework. The multi-SLAM framework is contained in a ROS node which subscribes to the laser data of the referring robots or sensors. A major difficulty in all rescue scenarios is limited communication, which aggravates the deployment of the multi-SLAM framework.

A reduced bandwidth leads to a reduced update rate of the robot pose which impedes the function of the motion controllers. However, as the amount of data exchanged contains only the laser and pose message, the traffic between the robots is comparably low.

5 Experiments

The experiments described in the following are hardly sufficient tests regarding a real rescue scenario. Nevertheless, the described approach aims at deployment at the Robocup rescue league which arena consists mainly of plain walls, such as the scenarios which where chosen for the described 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.

5.1 Single Robot Loop Closure Experiment

In this experiment our rescue robot “Simon” (Fig. 13a), equipped with a Hokuyo UTM-30LX LIDAR, was navigated through the first floor of a building at our campus. The test was aggravated through narrow 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 8192 cells with a granularity of 0.015 m. This results in total map edge length of 122.88 m.

Figure 4 shows three steps of the loop closing. A circle in Fig. 4 illustrates 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.

5.2 Single Robot SLAM with Reference Data

A second experiment deploying a single robot has been performed in order to test the SLAM framework. As input data, reference laser frames from a data repository at the University of Freiburg, Germany, were used [2].

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 6. Furthermore, the provided ground truth is compared to the estimated trajectory.

The data source of the University of Freiburg provides a valid ground truth. With also provided software, the generated localization can be compared with this reference data. The applied validation approach has been subject of Kuemmerle et al. [14].

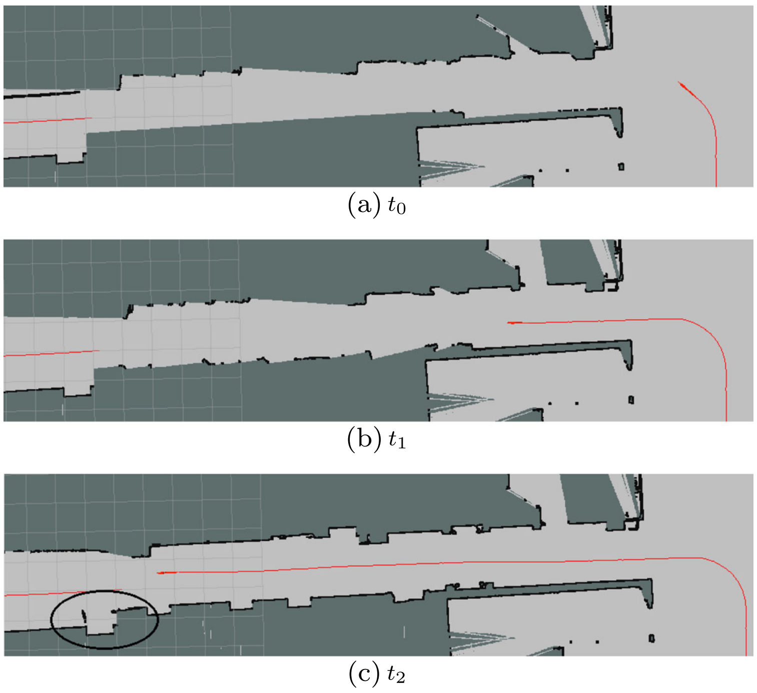


Fig. 4 Loop closing. Three steps of the final loop closing, chronological order from t_0 to t_2. A circle in c) marks a small loop closing error

(a) t_0

(b) t_1

(c) t_2

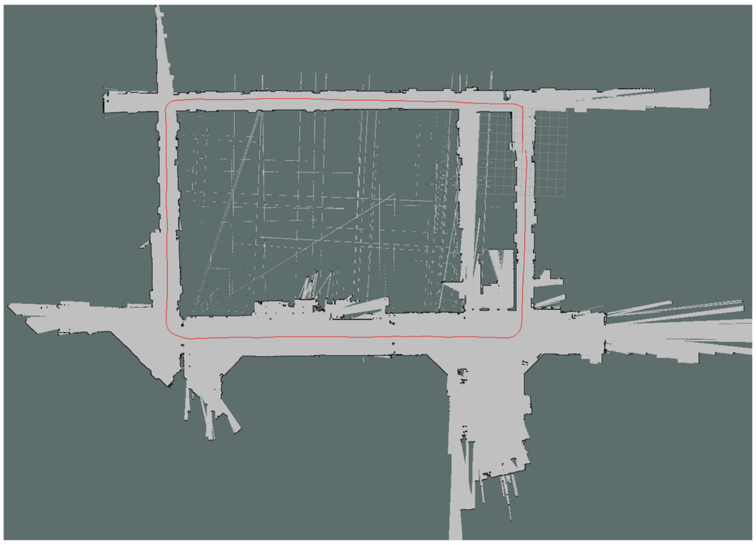


Fig. 5 Complete map of the single slam loop closure experiment with drawn trajectory

The generated error file consists of the angular and the linear error. Figure 7 depicts the data. The result shows a mean error in the translation of 0.046 m and 0.001 rad mean angular error.

The generated map and the results of the ground truth comparison show, that the multi-SLAM framework provides an accurate pose estimation. The mean error of the angle estimation is comparably low but the used data set contains a sizeable amount of straight transformations which contain only small angular errors. However, the maximal angular error of approximately 6 deg shows, that the pose estimation still needs improvement. The maximal translation error of roughly 0.2 m emphasizes this conclusion.

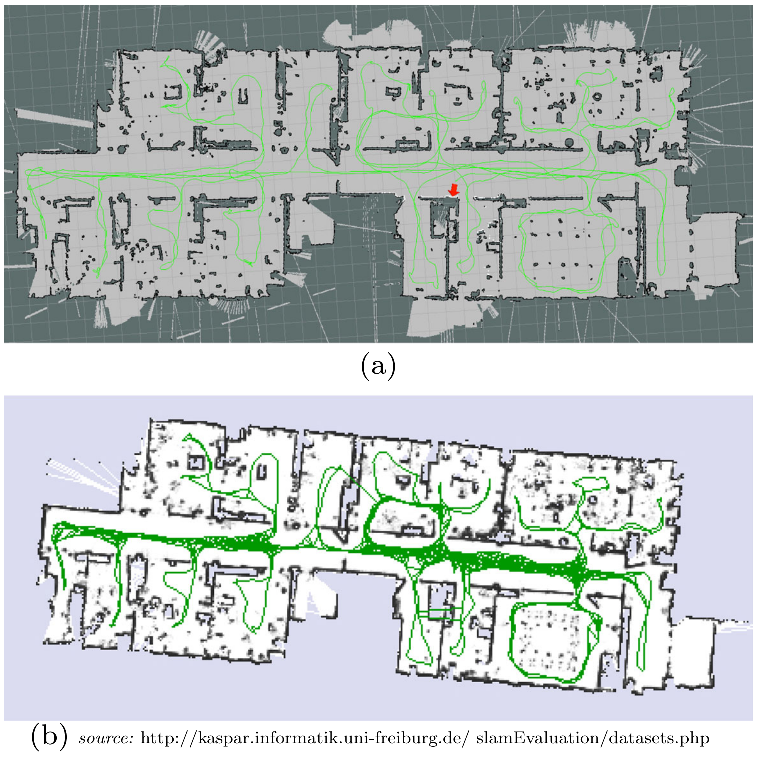
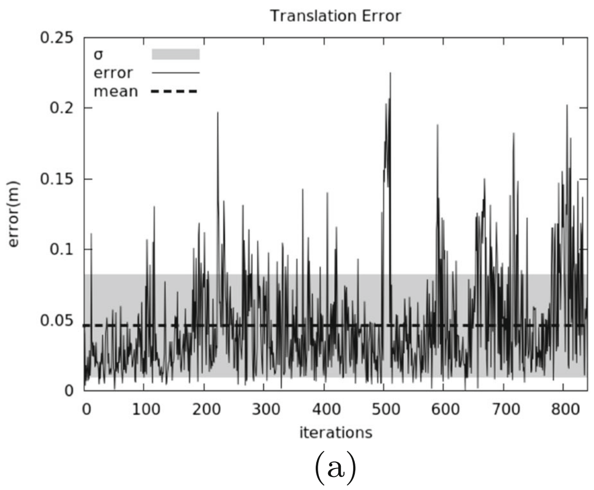

(a)
(b)
source: http://kaspar.informatik.uni-freiburg.de/ slamEvaluation/datasets.php

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



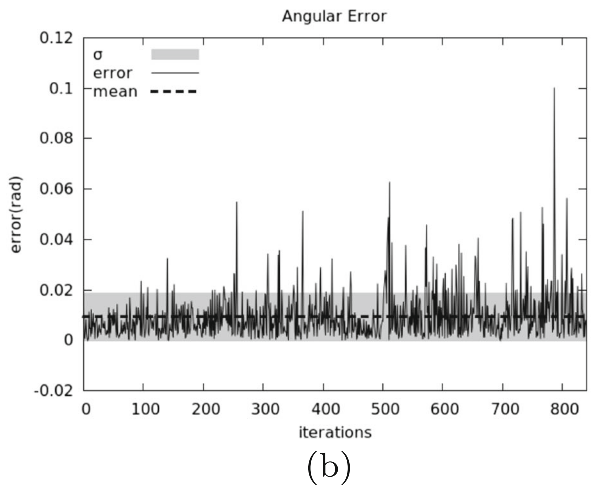


Fig. 7 Error Analysis of the reference data experiment. Figure **a** illustrates the error of the estimated translation, **b** shows a the error of the estimated angle

5.3 Single-SLAM with a Simulated Robot

The multi-SLAM framework has been tested with the ROS Simple Two Dimensional Robot Simulator (STDR).[1] The simulator provides artificial laser data for every robot, as well as an error-free ground truth. In this section, a single simulated robot was used to generate a map while a software recorded the ground truth of the simulator as well as the estimated pose of the SLAM. The map consists of 2048 cells with a granularity of 0.015 m.

A simple comparison of the localization and the ground truth would contain the global, accumulated error. Therefore, a different approach was used to validate the quality of the generated localization.

The program uses the ROS TF package to generate two arrays containing the the transformation of the ground truth $\mathbf{G} = \{\mathbf{g}_i \mid i = 1..n_g\}$ and the estimated pose of the SLAM $\mathbf{P} = \{\mathbf{p}_i \mid i = 1..n_p\}$ at equal timestamps.

To evaluate the generated error, two random indices i, j are generated. The algorithm determines the transformation $\mathbf{T}_{gij}$ of the ground truth and $\mathbf{T}_{pij}$ of the SLAM between the generated indices. The referring error is calculated by comparing $\mathbf{T}_{gij}$ and $\mathbf{T}_{pij}$.

The plots in Fig. 8 depict the results of this experiment. For better clarity, the single errors in this plot have been subsampled. The mean error of the translation is 0.020 m, the mean error of the estimated angle 0.011 rad.

The currently most widespread used 2D-SLAM approach in rescue robotics is Hector-SLAM ([12]). It features a high pose update rate and robust pose estimation. The approach uses few system resources and as their mapping is computed directly on an occupancy grid no conversion step as in the TSD SLAM approach is necessary. In order to compare the data of this section against the state of the art Hector-SLAM approach, it was deployed using similar parameters and the same laser input.

The plots depicted in Fig. 9 illustrate the calculated error of Hector-SLAM. For better clarity, the single errors in this plot have been subsampled. The mean error of the estimated translation is 0.015 m, the mean angular error is 0.005 rad. Figure 10 shows the generated maps of both slam approaches, as well as the estimated trajectory and the ground truth.

The data generated by this experiment shows, that the TSD SLAM approach is able to generate a comparably accurate pose estimation. However, as the comparison with the state of the art Hector-SLAM shows, the registration requires further improvement.

5.4 Multi-SLAM with Simulated Robots

In this section, the multi-SLAM framework is tested with the STDR simulator. The number of robots for this experiment has been set to the number of physical

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

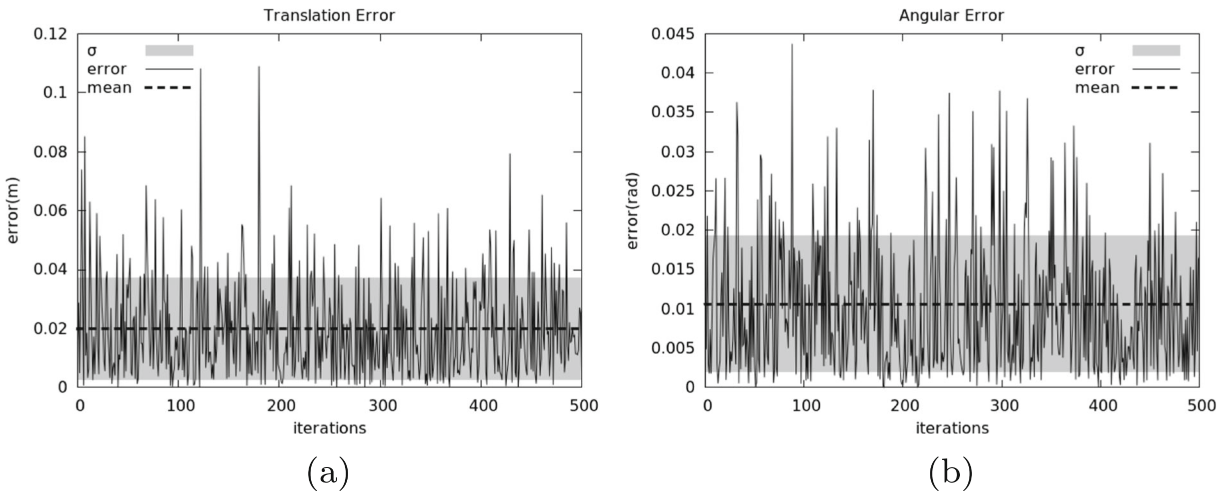


Fig. 8 Simulator TSD-SLAM Quality evaluation. Figure **a** Illustrates the error of the estimated translation, Figure **b** depicts the angular error

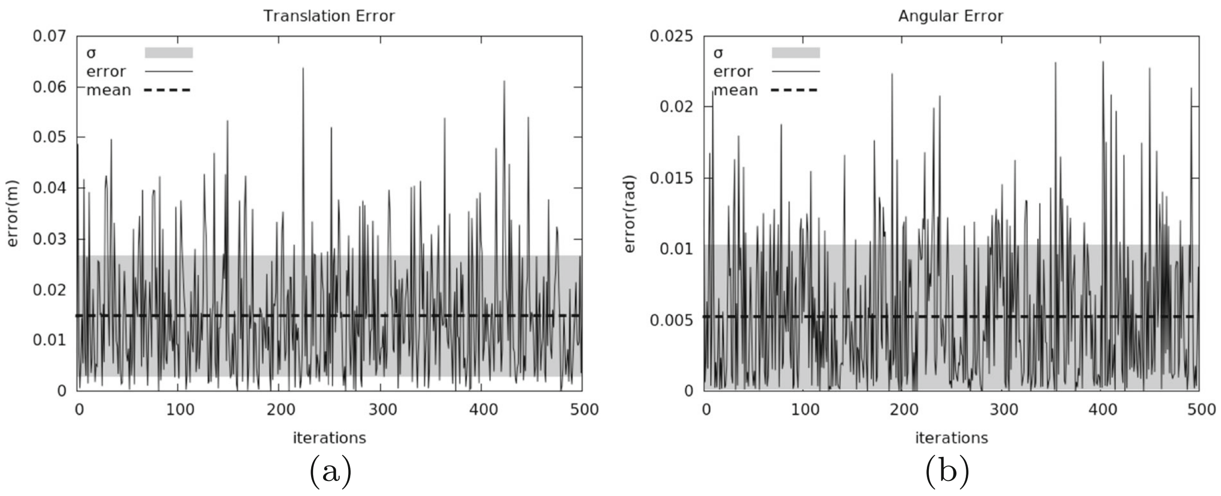
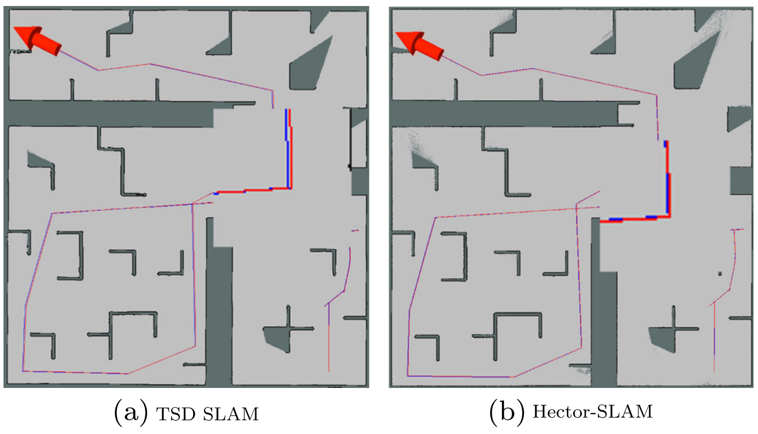


Fig. 9 Simulator Hector-SLAM Quality evaluation. Figure **a** illustrates the error of the estimated translation of Hector-SLAM, Figure **b** depicts the angular error

Fig. 10 Generated maps of the simulator. Figure **a** illustrates the mapping output of the TSD-SLAM, **b** the data of Hector-SLAM



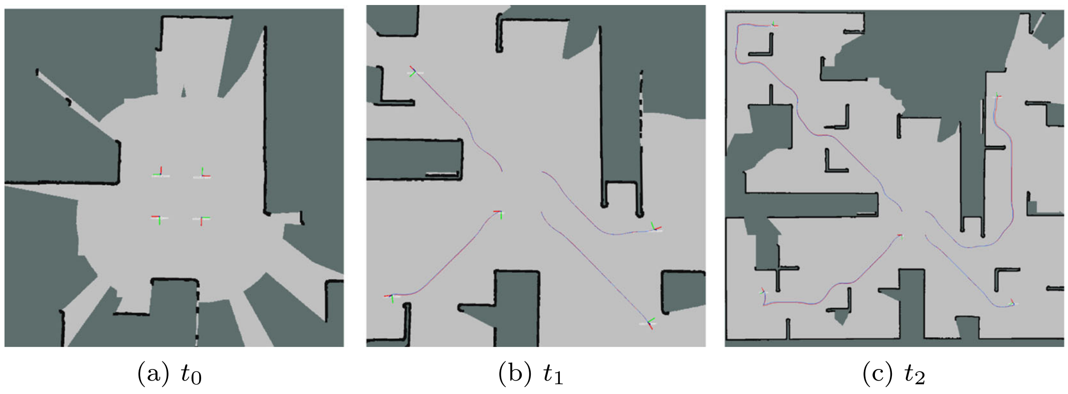


Fig. 11 Phases of the simulated multi-SLAM. Chronological order from t_0 to t_2

processor cores, wherefore hardware resources are used at high capacity. The map dimensions are the same as in the first experiment (Section 5.1).

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. Figures 11 and 12 show the process and the results of the simulated experiment. The blue line marks the ground truth, and the red line the estimated trajectory.

5.5 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 (Fig. 13) 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 5.1.

Figure 14 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. The final, simultaneously built map is displayed in Fig. 15.

During the multi-SLAM experiment, the timing of each thread has been logged in order to analyse this data. Figure 16 consists of plots depicting the timing. For better clarity, the single iteration times have been sub sampled. The localization threads show similar iteration times both at a mean of approximately 11 hz.

The mapping thread consumes a mean of 0.027 s. The iteration times of the occupancy grid are increasing as more data is added to the map, the referring mean is at 0.12 s. This process is obviously the most time consuming and gets more expensive as the map is filled. However, since the generation of the occupancy grid is only required occasional, it can be neglected.

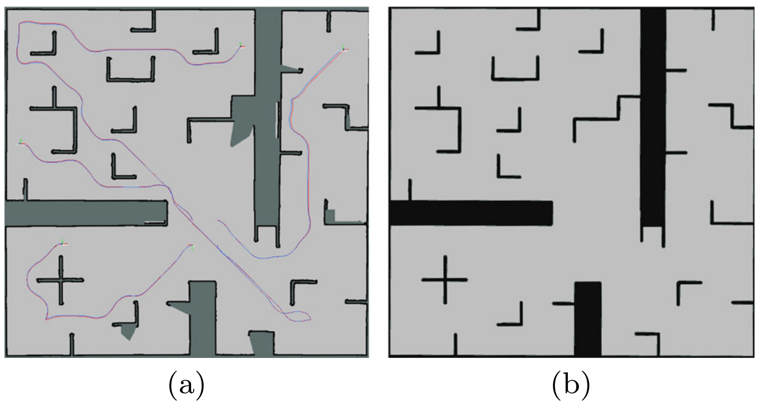


Fig. 12 Multi-SLAM simulation result. Comparison of the reconstructed map (**a**) and the original model used by the simulator (**b**)

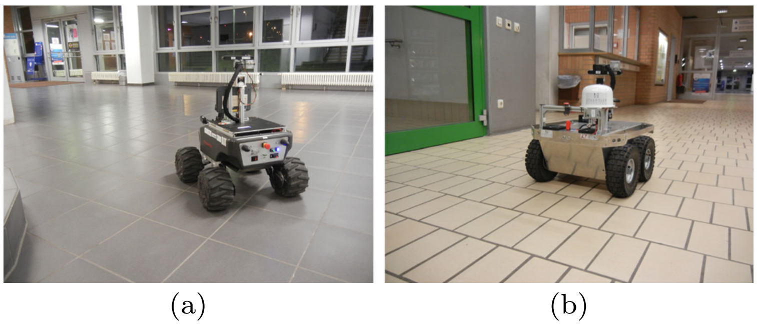


Fig. 13 Multi-SLAM with two cooperating robots. Image showing “Simon” (**a**) and “Georg” (**b**) during the multi-SLAM experiment

5.6 Compare to State of the Art Approach

As the intended use of the presented approach is a deployment at the Robocup rescue, the experiments in this section compare the TSD-SLAM against the most widespread used strategy, Hector-SLAM.

However, the referring ROS package does not provide multi-source-SLAM capabilities. Therefore, the following experiments compare the output of Hector SLAM to the TSD SLAM under the same conditions, using data sets of previously described experiments.

As the previous experiments in which this data has been used did not provide quantitative accuracy measurements due to the lack of a sufficient ground truth, this experiment does not contain such data as well. Therefore, the experiment data is supplied through screen shots of the generated maps. Similar to the previously described experiments, the loop closing error is the evaluation criteria,

Figure 17 shows the generated maps and the loop closing in detail of this experiment. Figure 17b, d, f show a loop closing error which shows, compared to the previous experiments, an advantage of the TSD SLAM, regarding comparably big maps.

6 Application at the Robocup German Open 2015

6.1 General

This section describes the deployment of the proposed approach at the Robocup rescue German Open 2015 competition. For the readers convenience, it includes a short summary of the most important rules.

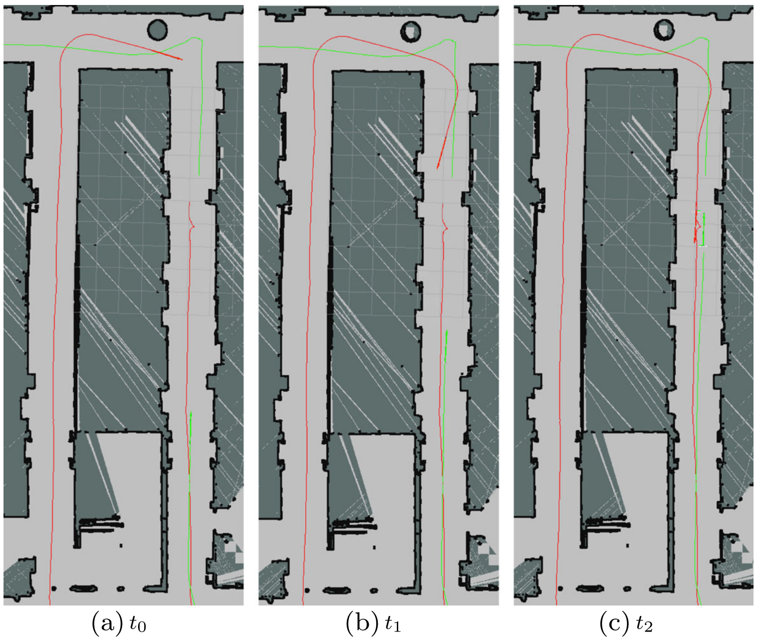


Fig. 14 Multi-SLAM loop closure. Image illustrating three steps of the robots closing their loops simultaneously from t_0 to t_2

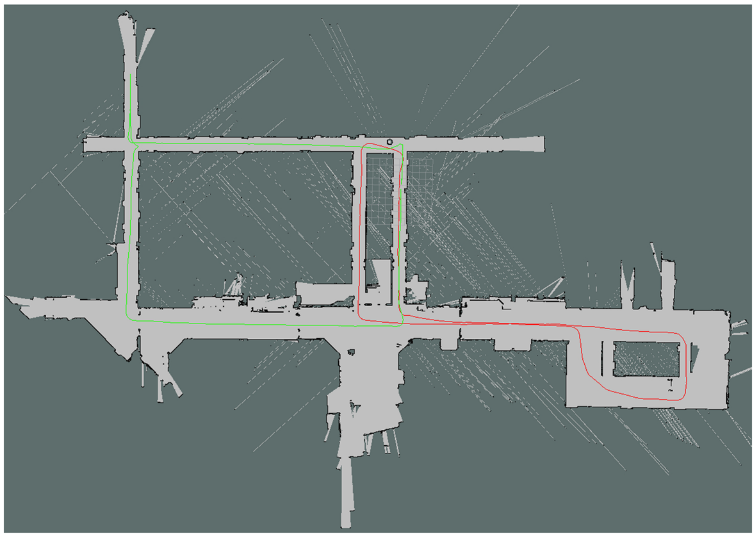


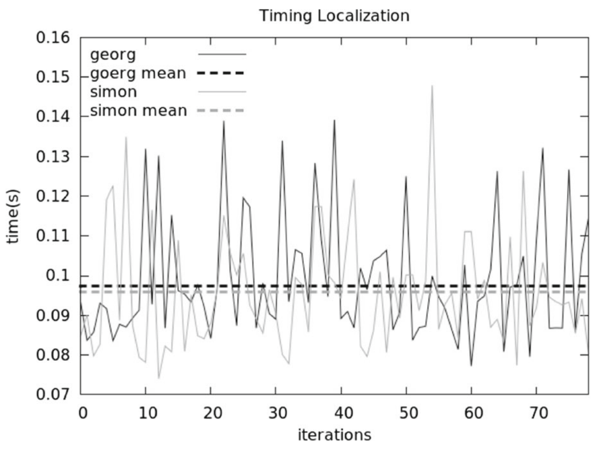
Fig. 15 Complete map of the cooperative mapping. Red color marks Simon's trajectory, green Georg's trajectory

Furthermore, special challenges regarding the competition are described, as well as necessary adaptations to the multi-SLAM framework, the results and the lessons learned.

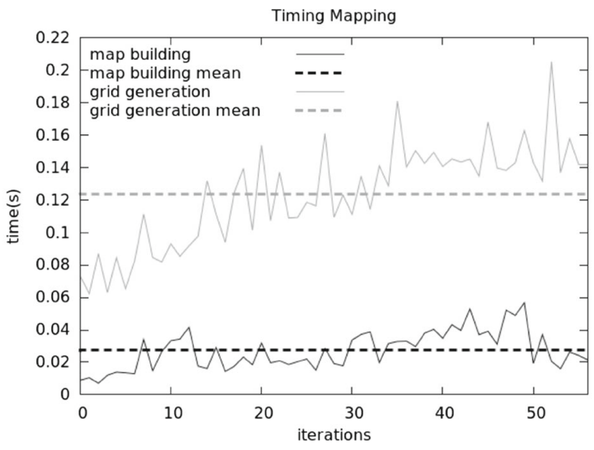
A team of robots has several upsides. At the Robocup, the number of cooperating robots in a team is infinite, but only one unit is permitted to be remote controlled. Our rescue robot team has therefore two members. A remote controlled robot with moderate mobility to explore the orange arena and an autonomous robot which is allowed to find victims in the yellow arena. The remote controlled robot also features a manipulator which allows better inspection of regions of interest.

Points are awarded for victim detection during mission by analyzing the robot's sensor data. The team also needs to provide data of the exploration after the run. This data, for instance the map or a list containing the victim positions, contains all information a robot collects.

However, as only a single version of each file or the map is rated, the data needs to be fused, for instance by stitching partial maps and applying gained transformations to the poses of the victims or other objects.



(a)



(b)

Fig. 16 Multi-SLAM timing analysis. This figure illustrates the timing of each thread. **a** Depicts the iteration times of the localization, **b** the iteration times of the mapping and the occupancy grid thread

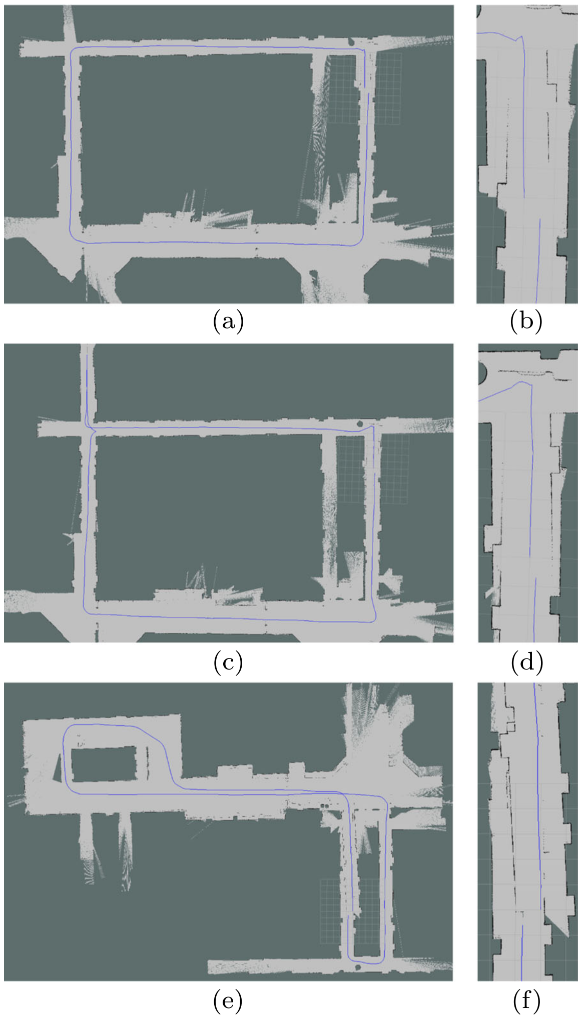


Fig. 17 Hector-SLAM experiment with data of chapters 5.1 and 5.5

If all robots build the same map simultaneously, as in our multi-SLAM approach, this is unnecessary. More details about the competition and the referring rules can be found on the homepage of the Robocup German Open 2015 rescue league [1].

6.2 Special Challenges

On the contrary to the experiments in Section 5, the arena at the competition simulates a disaster zone with different levels of obstacles. 2D SLAM was sufficient at the Robocup 2015, as the arena consisted mainly of plain walls. However, robots have to traverse 3D obstacles, such as ramps or beams. A LIDAR has to be kept levelled or the resulting map contains objects in wrong distances. A map could contain circular artefacts, for instance, if the laser scanner is pointing to the floor or the ceiling. At the Robocup rescue most of the teams use a platform to keep the laser levelled, and so does our team.

These platforms consist of two motors and an Inertial Measurement Unit (IMU) and have to compensate the roll- and tilt angle of the robot while being in motion. A malfunction of such a device aggravates the registration as the carried LIDAR points in a different angle to the wall. The registration of inconsistent data can lead to wrong localizations or wrongly placed objects in the map. Figure 18 displays a robot carrying a LIDAR on a levelling platform and the platform itself with more detail.

Even if the platform works within normal parameters some movements of the robots can exceed the dynamic capabilities of the levelling platform's soft- or hardware leading to the already described problems.

A tipped over robot can exceed the physical restrictions of the platforms as both motors have maximum angles defined by the mechanical design of the platform and the carried sensor. In this case, the sensor data has to be excluded from the registration.

Figure 19 depicts the described special challenges at Robocup rescue. Figure 19 illustrates the data of a wrong pointing LIDAR which results in map objects which are distant from the robot and should have been occluded by walls in front of the sensor.

Figure 19b shows the map at the same location in the arena as Fig. 18a. The crossing of such difficult obstacles results in fast movements of the robot which can exceed the dynamic capabilities of the levelling platform. This causes, as already mentioned, small registration errors which result in the displayed trajectory.

6.3 Adaptations

In order to react to these special conditions, the implemented software has been adapted. Especially to allow interaction with the SLAM. This is only necessary as both robots work on the same map. Taking a single SLAM approach into account, it can either be in working or in error state. In case of an error in a real scenario the robot would be lost. At Robocup, this is simulated with a hard reset which allows the team to

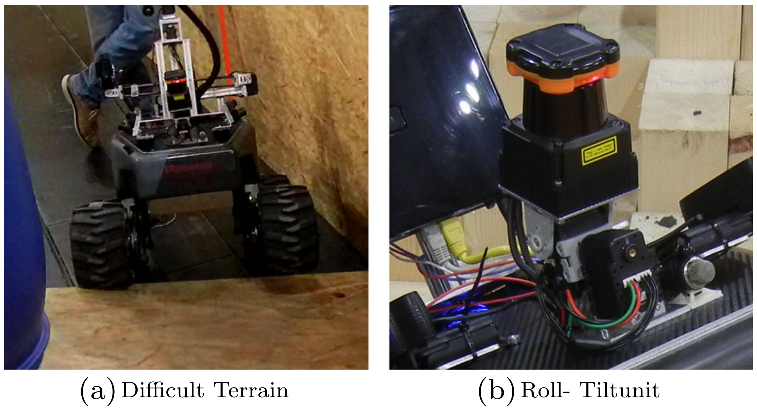


Fig. 18 Laser levelling. Image **a** showing one of our robot traversing a difficult obstacle in the arena, **b** depicts the roll- tilt unit carrying the LIDAR

restart the robot, in that case already scored points and collected data are lost.

Using cooperative SLAM, the remaining robots can carry on the mission, provided the lost robot does not destroy the map, for instance through a wrong pointing laser caused by tipping over. Therefore, the software has been extended with a communication interface which allows switching off the SLAM for each robot completely.

6.4 Cooperative Exploration Robocup Rescue

In this section, the deployment of the multi-SLAM framework at the Robocup Rescue 2015 is depicted. In order to validate the capabilities of our approach under the circumstances of a competition, images of cooperatively acquired maps and trajectories are provided.

Figure 20 illustrates the cooperative exploration in the arena. The red trajectory is generated by the autonomous robot which stays in the yellow arena. As there are only moderate obstacles in this part, the map shows few errors. The blue trajectory refers to the remote controlled robot which explores the orange part of the arena which contains more complex obstacles such as wooden beams or the already mentioned crossed ramps.

The final map shows a loop closing error resulting from minor to moderate registration errors caused by the dynamic movement whilst crossing complex obstacles. Bigger registration errors, for instance caused by a not correctly levelled LIDAR, result in wrong objects which in some cases should have been occluded by other mapped objects.

6.5 Lessons Learned

On the one hand, the provided data shows that our multi-robot-SLAM approach works well under the conditions of a competition. The advantage over single robot-SLAM is clear, as we do not need to combine the data of the cooperating robots after the mission.

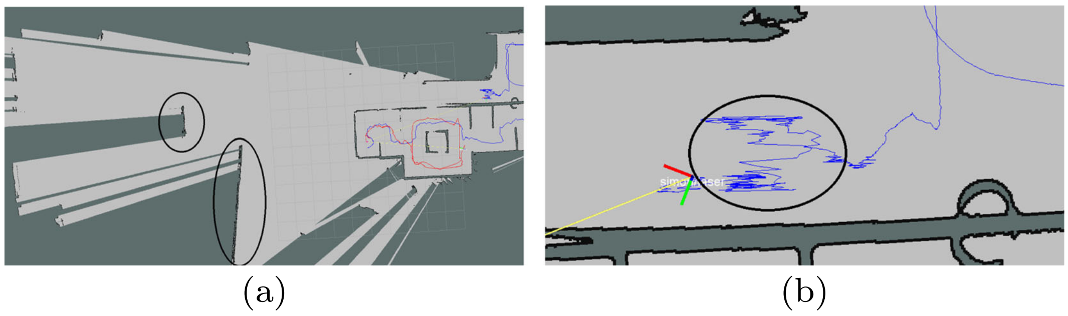


Fig. 19 Special SLAM challenges. Figures illustrating two possible problems caused by difficult obstacles (marked with circles). **a** Wrong pointing LIDAR resulting in objects distant from the robot which should have been occluded. **b** Difficult to traverse obstacle resulting in dynamic movements of the robot leading to small registration errors

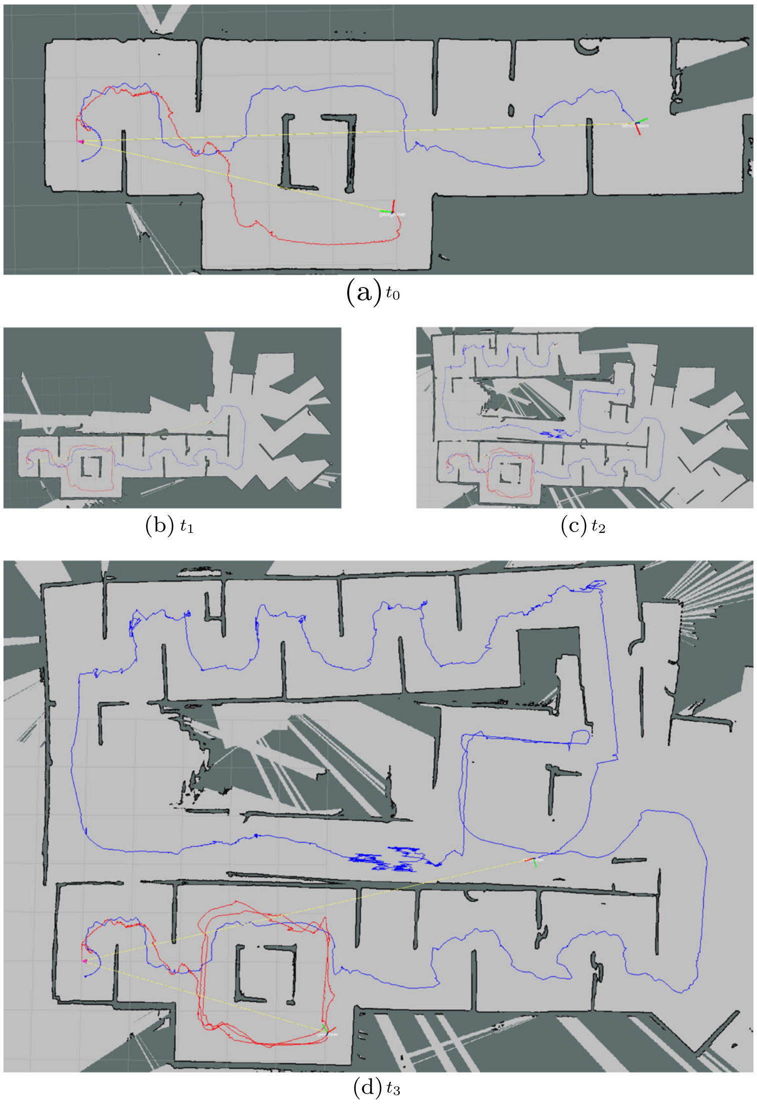


Fig. 20 Cooperative SLAM Robocup rescue. Four steps of the cooperative SLAM performed at the Robocup (from t_0 to t_3). Both robot start in the same pose, red trajectory refers to the autonomous robot, blue to the remote controlled

On the other hand however, the generated maps show how complex obstacles and the resulting dynamic movement of the robot aggravate the registration. In order to deal with fast pose changes of the robot, an improved registration algorithm is necessary which can cope with less overlapping areas in the sensor frames. Moreover, the laser levelling platform has to be improved in order to deal with these fast pose changes.

Another issue which is unique to the deployment of an SDF approach in the Robocup rescue competition results in disappearing objects, eg. walls. This problem is depicted in Fig. 21. As an object is represented by a sign change in the SDF, it consists of several cell layers of positive and negative SDF values (Fig. 2c). The walls at the Robocup arena have a thickness of merely 5 cm and are sometimes visible from both sides.

Figure 21 illustrates the described problem. A disappearing wall is a severe problem for localization and robot navigation. As this problem is currently not sufficiently explainable, it needs to analyzed further in future work. In this case, Hector SLAM has advantages. Figure 21d illustrates the map output of their approach using the same data under the same conditions, which does not have the same problem.

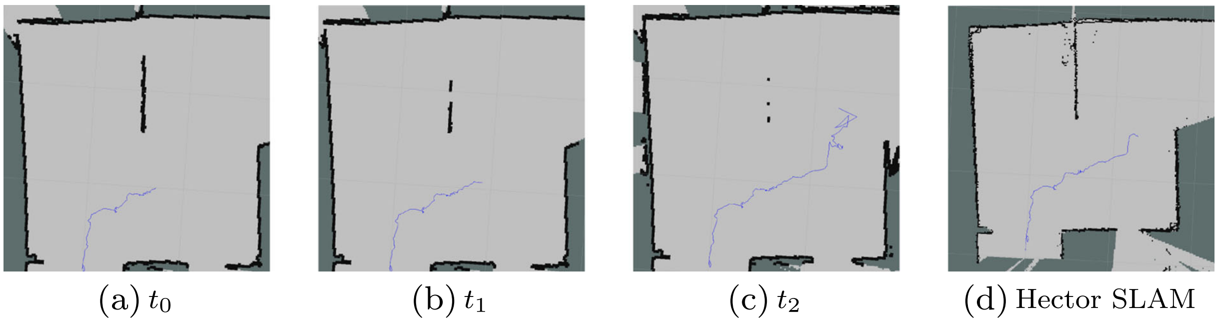


Fig. 21 Disappearing walls. Image illustrating how a wall seen from both sides disappears from t_0 to t_2. c) Hector-SLAM with the same input data

7 Conclusion and Future Work

In this publication, we presented a multi-SLAM framework based on SDF. We illustrated how our previous work [15] is extended to a simultaneous multi source localization and mapping application. The provided experiments showed its capabilities and the deployment at the Robocup rescue competition has proven its advantages over a single-SLAM system.

The comparison of TSD SLAM against Hector-SLAM revealed, that the multi-SLAM framework has advantages on bigger maps. However, as the ground truth experiments and the Robocup competition showed, the registration module has weaknesses. Future work will therefore focus on new registration techniques.

Moreover, the reason for the disappearing thin walls, which caused problems in the competition, needs to be evaluated. Limited or jammed communication is a problem in real rescue scenarios. The multi-SLAM uses a comparably low amount of data traffic but, nevertheless, future work will also consist of experiments determining the minimal bandwidth necessary.

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

References

1. Robocup rescue german open 2015. https://www.robocupgermanopen.de/en/major/rescue. Online; accessed 15-November-2015
2. slam benchmarking. http://kaspar.informatik.uni-freiburg.de/slamEvaluation. Online; accessed 14-January-2015
3. Burgard, W., Moors, M., Fox, D., Simmons, R., Thrun, S.: Collaborative multi-robot exploration. In: IEEE International Conference on Robotics and Automation, 2000. Proceedings. ICRA '00, vol. 1, pp. 476–481 (2000). doi:10.1109/ROBOT.2000.844100
4. Burgard, W., Moors, M., Stachniss, C., Schneider, F.: Coordinated multi-robot exploration. IEEE Trans. Robot. **21**(3), 376–386 (2005)
5. Chen, Y., Medioni, G.: Object Modeling by Registration of Multiple Range Images. In: 1991 IEEE International Conference On Robotics and Automation, 1991. Proceedings., vol. 3, pp. 2724–2729 (1991)
6. Fox, D., Ko, J., Konolige, K., Limketkai, B., Schulz, D., Stewart, B.: Distributed Multi-Robot Exploration and Mapping. In: Proceedings of the IEEE, p. 2006 (2006)
7. Granstrom, K., Callmer, J., Ramos, F., Nieto, J.: Learning to Detect Loop Closure from Range Data. In: IEEE International Conference On Robotics and Automation, 2009. ICRA '09, pp. 15–22 (2009)
8. Howard, A.: Multi-Robot Simultaneous Localization and Mapping Using Particle Filters. In: Proceedings of Robotics: Science and Systems, Cambridge, USA (2005)
9. Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., Fitzgibbon, A.: 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. Kim, B., Kaess, M., Fletcher, L., Leonard, J., Bachrach, A., Roy, N., Teller, S.: Multiple Relative Pose Graphs for Robust Cooperative Mapping. In: IEEE International Conference on Robotics and Automation, ICRA, pp. 3185–3192 (2010)
11. Koch, P., May, S., Schmidpeter, M., Kuhn, M., Pfitzner, C., Merkl, C., Koch, R., Fees, M., Martin, J., Nuchter, A.: Multi-robot localization and mapping based on signed distance functions. In: 2015 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), pp. 77–82 (2015). doi:10.1109/ICARSC.2015.18
12. Kohlbrecher, S., Meyer, J., von Stryk, O., Klingauf, U.: A Flexible and Scalable Slam System with Full 3D Motion Estimation. In: Proceedings IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR), IEEE (2011)

13. Konolige, K., Fox, D., Ortiz, C., Agno, A., Eriksen, M., Limketkai, B., Ko, J., Morisset, B., Schulz, D., Stewart, B., Vincent, R.: Centibots: Very Large Scale Distributed Robotic Teams. In: Khatib, M.HA.Jr., O. (ed.) ISER, Springer Tracts in Advanced Robotics, vol. 21, pp. 131–140. Springer (2004)
14. Kümmerle, R., Steder, B., Dornhege, C., Ruhnke, M., Grisetti, G., Stachniss, C., Kleiner, A.: On measuring the accuracy of slam algorithms. Auton. Robot. **27**(4), 387–407 (2009)
15. May, S., Koch, P., Koch, R., Merkl, C., Pfitzner, C., Nüchter, A.: 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, pp. 95–102 (2014)
16. Osher, S., Fedkiw, R.: Level Set Methods and Dynamic Implicit Surfaces (Applied Mathematical Sciences). 2003rd edn. Springer (2002)
17. Zhang, Z.: Iterative point matching for registration of free-form curves and surfaces. Int. J. Comput. Vis. **13**(2), 119–152 (1994)

Philipp Koch was born in Göttingen, Germany, in 1982. He graduated with a Bachelor of Engineering at the Nuremberg Institute of Technology (NIT) in the field of electrical engineering and information technologies in 2012. In the year 2014, he received the degree Master of Science in the field of 2D/3D reconstruction and SLAM, also at the NIT. Since 2014, he works as a research assistant at the NIT in the field of mobile robotics, focussing on autonomous systems in the field of industrial intralogistic robots and rescue robotics. His research work includes 2D/3D reconstruction, SLAM and traversability.

Stefan May received a diploma (electrical engineering) in 2000 and a Masters degree (software engineering) in 2004 from the Georg Simon Ohm University of Applied Sciences in Nuremberg. In 2009 he received his PhD in computer vision from the University in Osnabrueck. Between 2002 and 2009 he was working in the context of test systems for automotive electronics and 3D vision for mobile robotics at Audi and Fraunhofer IAIS in Germany and INRIA in France. Since 2010, he is professor for Automation and Mechatronics at the Technische Hochschule Nuremberg Georg Simon Ohm. He is head of the Laboratory for Mobile Robotics focusing academic and industrial applications. His main research interests include sensors and data processing for 3D perception of mobile robots.

Michael Schmidpeter was born in 1989. He received his Bachelor of Engineering degree in electrical engineering and information technologies in 2013 at the Nuremberg Institute of Technology (NIT). In 2015, he graduated with a Master of Science in the field of autonomous robots, also at the NIT. Since 2015 he works as a research assistant at the Nuremberg Campus of Technology. His main focus lies on multi-robot navigation.

Markus Kühn was born 1992 in Fürth, Germany. In 2014, he received his Bachelor of Engineering degree in computer science at the Corporate State University of Baden-Wuerttemberg in Friedrichshafen, Germany. He graduated at the Nuremberg Institute of Technology (NIT) with a Master of Science degree in 2016 which focused on applied robotics research. In particular, he studies approaches for localization and mapping as well as software architectures for drone autonomy.

Christian Pfitzner was born in 1987 in Nuremberg, Germany. He received the Bachelor of Engineering degree in electrical engineering and information technologies at the Nuremberg Institute of Technology (NIT), Germany, in 2011. For the Master of Science degree he studied exploration strategies for mobile robotics in rescue environment, also at the NIT. Since 2014 he is a PhD student at the Graduate School of Live Sciences at the Julius-Maximilians University Wuerzburg, Germany. Since 2014 he works as a research assistant at the Nuremberg Campus of Technology in the field of mobile robotics and 3d vision for medical applications. Furthermore, he is a lecturer for the field of mobile robotics at the NIT and the ROS Summer School in Aachen, Germany. Website: christian-pfitzner.de/research.

Christian Merkl was born in 1983 in Nuremberg, Germany. He received the Bachelor of Engineering degree in electrical engineering and information technologies at the Nuremberg Institute of Technology (NIT), Germany, in 2012. For the Master of Science degree he studied detection of the human body using contactless sensor systems, also at the NIT. Since 2014 he works as a research assistant at the Nuremberg Campus of Technology in the field of HMI. Furthermore, he is a temporary lecturer for computer science.

Rainer Koch received his diploma (power engineering and system control) in 2007 and his Master‘s degree (electronical and mechatronical systems) in 2010 from the Georg-Simon Ohm University of Applied Science in Nuremberg. Between 2007 and 20011 he was working as a research assistant at the Institute for Power Electronics ELSYS at the Georg-Simon Ohm University of Applied Science. Since 2011 he is a PhD student at the autonomous robotics team at the Nuremberg Institute of Technology Georg-Simon Ohm. His research interests include sensor and data processing for 2D and 3D perception of mobile robots.

Martin Fees was born 1991 in Erlangen, Germany. He received the Bachelor of Engineering degree in mechanical engineering 2014 at the Nuremberg Institute of Technology (NIT), Germany. Since 2014 he is receiving his Master of Science degree by developing an inspection and manipulation arm for mobile rescue robots at the NIT.

Jon Martin was born in Bilbao, Spain, in 1989. He first graduated as a technical industrial engineer, specialized in Electronics at the University of the Basque Country in 2011. Furthermore, he received a Diploma in Electronics and Automatization in also at the University of the Basque Country in 2014. He did his Thesis as an exchange student at the Nuremberg Institute of Technology, it included navigation and path-path planning of a mobile robot in an industrial environment. Since then he works as a research assistant at the NIT. His main focus lies on indoor service robotics and multi robot communication.

Daniel Ammon was born in Nuremberg, Germany, in 1988. He graduated with a Bachelor of Engineering at the Nuremberg Institute of Technology (NIT) in the field of mechatronics with priority on automation, in 2015. Since then, he studies as a master's degree candidate at the Nuremberg Institute of Technology in the field of mobile rescue and service robotics. His main focus lies on localization algorithms for indoor service robotics. His actual work includes the development of a particle filter algorithm for indoor localization and improvements on 2D SLAM algorithms.

Andreas Nüchter is professor of Computer Science (telematics) at University of Würzburg. Before summer 2013 he was affiliated with the Automation group at Jacobs University Bremen, with the University of Osnabrück, and with the Fraunhofer Institute for Autonomous Intelligent Systems (AIS, Sankt Augustin). Prof. Nüchter holds a PhD and a diploma degree in Computer Science from the University of Bonn. His diploma thesis was awarded by the German society of informatics (GI) and his dissertation was shortlisted for the EURON PhD award. Prof. Nüchter works on telematics systems for robotics and automation, 3D vision and laser scanning, cognitive systems and artificial intelligence.