Large Scale 2D Laser SLAM using Truncated Signed Distance Functions

Kevin Daun[1], Stefan Kohlbrecher[1], Jürgen Sturm[2] and Oskar von Stryk[1]

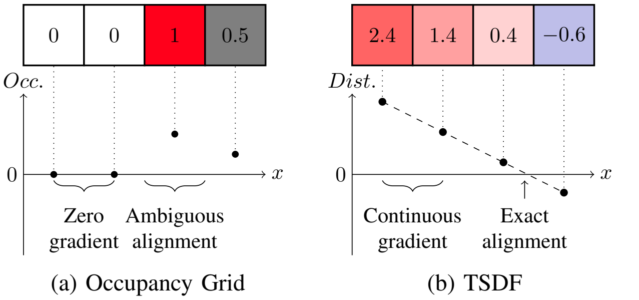
Abstract— **For deployment in previously unknown, unstructured and GPS-denied environments, autonomous mobile rescue robots need to localize themselves in the environment and create a map of it using a simultaneous localization and mapping (SLAM) approach. While most existing lidar-based methods use occupancy grids to represent a map, the use of truncated signed distance functions (TSDFs) is investigated in this paper to improve accuracy and robustness. In contrast to occupancy grids, TSDFs represent the distance to the nearest surface in every grid cell. This enables sub-pixel precision during localization and increases the basin of convergence of scan matching. To enable consistent mapping of large spaces, an efficient branch-and-bound based loop closure detection is applied. The evaluation of the proposed approach with publicly available benchmark data shows that the proposed approach yields improved accuracy in comparison to occupancy grid based methods, while requiring similar runtime. Furthermore, it is demonstrated that the proposed approach is able to map a large scale environment with urban search and rescue elements in real-time.**

I. INTRODUCTION

To perform missions within large, unknown and degraded environments, autonomous mobile rescue robots need to localize themselves in such environments and create a map of it using a simultaneous localization and mapping (SLAM) approach. The ability to create such a map and locate the robot's pose in it are a key prerequisite for many higher level autonomous functions such as navigation or exploration.

Current state-of-the-art 2D laser SLAM systems such as Cartographer [1], Hector SLAM [2] or GMapping [3] represent maps using occupancy grids where every map cell represents the probability of being occupied. As a result, the localization accuracy of the occupancy grids is inherently limited by the map resolution (see Fig. 1). An alternative to occupancy grid maps are truncated signed distance functions (TSDFs) [4] where every cell models the distance to the nearest object surface enabling sub-pixel accuracy. As TSDFs provide gradients around the surface they are naturally suited for optimization based approaches. Furthermore, since meaningful gradients exist in a larger area, scan matching is expected to be more robust to poor initialization.

TSDF-based approaches became popular with RGB-D cameras [5], but so far their utility for laser-based 2D SLAM has not been fully exploited. The main challenge in applying these methods to laser scanners is, that those sensors also provide distance readings under highly slanted angles, which leads to poor surface reconstruction. This requires different update and inference schemes. Recently, 2D TSDF SLAM approaches have been explored. Koch et al. [6] applied TSDFs to urban search and rescue (USAR) environments demonstrating reduced drift. Fossel et al. [7] demonstrated improvements in accuracy for office like environments. However, these methods are not able to map large environments as they do not detect loop-closures. In this paper, we present a full SLAM system based on TSDFs and show in experiments that it outperforms occupancy grids both in terms of accuracy and robustness.



(a) Occupancy Grid (b) TSDF

Fig. 1: 1D comparison of a TSDF and an occupancy grid with an ideal sensor. The TSDF accurately reconstructs the position of the surface whereas the accuracy of occupancy grid is limited by the grid resolution. As the TSDF provides gradients in a larger area around the surface, it yields a larger convergence radius for scan matching.

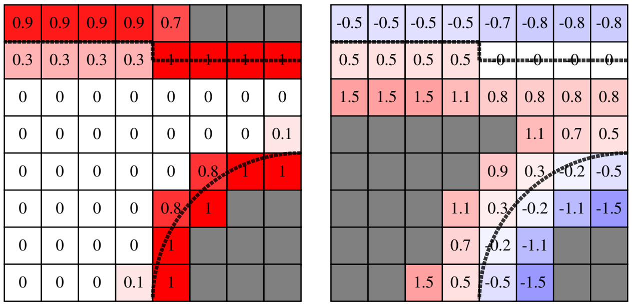
We base our work on Cartographer, an open-source SLAM system that implements scan-to-map matching and loop closure detection. We extend Cartographer to support TSDFs as an alternative map representation, and introduce corresponding algorithms for scan matching and loop closure detection. The main contributions of this paper are:

- Investigation of TSDF update rules for long-range sensors
- Extension of the branch-and-bound loop closure detection [1] for TSDFs
- Quantitative evaluation and comparison of TSDF and Occupancy Grid based-methods

Our evaluation on artificial data sets and publicly available benchmark data shows that the proposed approach yields improved accuracy in comparison to occupancy grid based methods while improving robustness and requiring similar runtime. Furthermore, we demonstrate that the proposed approach is able to map a large scale environment with USAR elements in real-time. Our implementation is publicly

[1] Simulation, Systems Optimization and Robotics Group, Technische Universität Darmstadt, Hochschulstr. 10, 64289 Darmstadt, Germany {daun, kohlbrecher, stryk}@sim.tu-darmstadt.de

[2] Google Germany GmbH, Erika-Mann-Str. 33, 80636 München, Germany jsturm@google.com



0.9	0.9	0.9	0.9	0.7			
0.3	0.3	0.3	0.3	1	1	1	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0.1
0	0	0	0	0	0.8	1	1
0	0	0	0	0.8	1		
0	0	0	0	1			
0	0	0	0.1	1			

(a) Occupancy Grid

-0.5	-0.5	-0.5	-0.5	-0.7	-0.8	-0.8	-0.8
0.5	0.5	0.5	0.5	0	0	0	0
1.5	1.5	1.5	1.1	0.8	0.8	0.8	0.8
					1.1	0.7	0.5
				0.9	0.3	-0.2	-0.5
			1.1	0.3	-0.2	-1.1	-1.5
			0.7	-0.2	-1.1		
		1.5	0.5	-0.5	-1.5		

(b) TSDF

Fig. 2: A two-dimensional occupancy grid and TSDF. The underlying surface of the scene is indicated by the dashed lines. Undefined cells are marked in gray.

available[1].

II. Related Work

The SLAM problem covers two aspects – estimating the robot pose and mapping the environment. This requires suitable techniques to model the environment, which influence the choice of methods to optimize the pose and the map.

Common paradigms to address the SLAM problem are particle filter [3] and graph-based methods [8].

Graph based approaches, introduced by Lu and Milios [9], combine a front-end and back-end [10]. The front-end continuously estimates the pose based on local observations such as odometry or scan matches. The back-end incorporates global observations such as information from revisiting poses and continuously updates the whole pose history. The robot trajectory and map are modeled as a graph, consisting of nodes for each pose which are constrained by measurements.

While optimizing a pose graph in real-time is usually possible, detecting constraints requires an exhaustive search in a large space. Therefore, Hess et al. [1] propose a branch-and-bound approach for computing loop closure constraints in occupancy grids in real-time. Occupancy grids (see Fig. 2a) represent the environment as a volumetric grid, assigning each cell a probability of being occupied.

Focusing on robustness against rough environments, Kohlbrecher et al. [2] propose Hector SLAM, a front-end maintaining a 2D map with 3D motion estimation. It applies multi-resolution occupancy grids to achieve robust scan matching.

A method to improve the robustness of scan matching to poor initialization is real-time correlative scan matching [11] which efficiently samples poses around the initial scan matching pose to avoid local optima.

Droeschel et al. [12] propose to use local multi-resolution grids to improve the computational efficiency for mapping with laser scanners.

In their seminal work, Curless and Levoy [4] propose modeling the environment using TSDFs (see Fig. 2b). This method became popular with RGB-D sensors and KinectFusion [5] an approach capable of live 3D tracking and reconstruction. Performing a point-to-plane iterative closest point algorithm (ICP) optimized for efficient usage of GPU parallelization they were able to generate high-resolution 3D maps in real-time. Bylow et al. [13] propose to directly minimize the depth error of the RGB-D image on the TSDF. Thereby, they demonstrate improvements in accuracy in comparison to KinectFusion. Slavcheva et al. [14] propose representing the scan as TSDF and performing direct TSDF to TSDF matching and demonstrate further improvements in the size of the convergence basin, rotational motion estimation and reconstruction fidelity.

LiDAR systems have a larger range in comparison to RGB-D cameras, leading to more measurements with steep incident angles and thereby increasing projection errors of TSDFs. Fossel et al. [7] propose a 2D laser TSDF front-end approximating the scan with regression lines to correct for the projection error and show improved accuracy in comparison to occupancy grids.

Koch et al. [6] propose ohm_tsd_slam, an ICP based 2D laser TSDF front-end enabling multi-robot SLAM in USAR environments.

Our approach applies a direct optimization of scan poses based on minimizing the distance values in the TSDF [13]. To improve robustness to poor initialization of scan matching we apply real-time correlative scan matching [11]. We investigate and compare two update schemes for the TSDF: projective distance [5] and an approximation of the Euclidean distance based on normal projections which is related to [7]. To enable the mapping of large areas with loop closure in real-time with TSDFs, we propose a loop closure method for TSDFs which is derived from the branch-and-bound based method for occupancy grids in [1].

III. Approach

Following [1], we split the the SLAM problem into a local and a global optimization phase.

Local SLAM (or SLAM frontend) computes locally consistent submaps by performing scan-to-map matching and is thereby able to generate a pose estimate and a map in real-time. Scan matching induces small errors, leading to drift in the pose estimate over time.

To maintain global consistency of the map, Global SLAM subsequently optimizes the poses of all submaps and scans in a pose graph. Whenever a submap is finished, the constraint detection checks for loop closures and adds the new constraints to the pose graph. Optimizing the pose graph yields a globally consistent map.

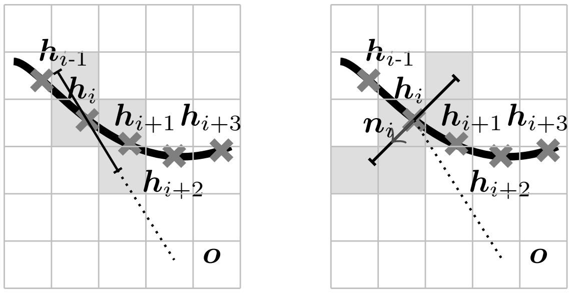
IV. Local SLAM

Local SLAM solves the online SLAM problem. First, the sensor pose is estimated by aligning the latest laser scan with the map. Afterwards, the map is updated to incorporate the scan information.

A. Scans

We denote a single scan $\boldsymbol{H} = \{\boldsymbol{h}_i\}_{i=1,\dots,N}$ consisting of N range observations $\boldsymbol{h} = (h_x, h_y)$ in the scan coordinate

[1] https://github.com/tu-darmstadt-ros-pkg/cartographer/tree/tsdf



(a) Projective Distance Update (b) Approximate Euclidean Distance Update

Fig. 3: To update the TSDF with a scan observation $\boldsymbol{h}_i$ we evaluate two distance methods. (a) updates the cells in direction of measurement $\boldsymbol{h}_i$ to origin $\boldsymbol{x}$. (b) updates the cells in direction of the scan normal $\boldsymbol{n}_i$.

frame with the sensor pose as the origin of the scan frame. The rigid transformation $\boldsymbol{T}_{\boldsymbol{\xi}}$, with $\boldsymbol{\xi} = [\xi_x, \xi_y, \xi_\theta]^T$ transforms a point $\boldsymbol{p} \in \mathbb{R}^2$ from the scan frame into the map frame and is defined as

$$\boldsymbol{T}_{\boldsymbol{\xi}}\boldsymbol{p} = \boldsymbol{R}_{\boldsymbol{\xi}}\boldsymbol{p} + \boldsymbol{t}_{\xi} = \begin{pmatrix} \cos\xi_\theta & -\sin\xi_\theta \\ \sin\xi_\theta & \cos\xi_\theta \end{pmatrix} \boldsymbol{p} + \begin{pmatrix} \xi_x \\ \xi_y \end{pmatrix}. \quad (1)$$

B. Maps

The key difference of our approach to [1] is that we model the map as a TSDF. The Signed Distance Function $\Phi : \mathbb{R}^2 \to \mathbb{R}$ represents the scalar, signed distance to the nearest surface for each position in space. Φ is positive outside of objects and negative inside of objects. Therefore, object surfaces are encoded as the zero isocontour ($\Phi = 0$). Instead of evaluating Φ completely, TSDFs evaluate Φ only close to the surface. Each evaluation of a position further away from the closest surface than the truncation distance τ is truncated

$$\Phi_\tau(\boldsymbol{x}) = \begin{cases} -\tau, & \text{if } \Phi(\boldsymbol{x}) < -\tau \\ \Phi(\boldsymbol{x}), & \text{if } |\Phi(\boldsymbol{x})| \leq \tau \\ \tau, & \text{if } \Phi(\boldsymbol{x}) > \tau. \end{cases} \quad (2)$$

To represent a 2D scene as a TSDF, space is discretized in a regular grid. Each grid cell c contains the current estimate of the TSDF $M_\Phi(c)$, and a scalar weight $M_w(c)$ indicating the confidence in the TSDF value.

1) Projective Distance Update: Estimating the proper Euclidean distance for every cell is computationally intense. Therefore, most approaches approximate it. A common approach for updating the TSDF from depth images is the projective distance cell updates [5] (see Fig. 3a). To update a TSDF from a new range observation, we model each measurement as a ray from the sensor origin located at the origin of the sensor frame to the position of the measurement $\boldsymbol{h}_i$. Thus, the direction is $\hat{\boldsymbol{h}} = \frac{\boldsymbol{h}_i}{||\boldsymbol{h}_i||}$. We parameterize the scan based on an interpolation parameter u as a ray

$$\boldsymbol{v}_{projective}(u) = \boldsymbol{h}_i + u\hat{\boldsymbol{h}}. \quad (3)$$

All cells c along the ray $\boldsymbol{v}(u)$ with $u \in [-||\boldsymbol{h}_i||, \tau]$ are updated by taking a weighted moving average of the distance measurements

$$M_\Phi(c) := \frac{M_W(c) M_\Phi(c) + \omega(u)\Phi_\tau(u)}{W(c) + \omega(u)} \quad (4)$$

$$M_W(c) := M_W(c) + \omega(u) \quad (5)$$

with the update weighting function $\omega(u)$. We apply an exponential weighting function as proposed in [13]. Additionally, we reduce the weight outside the truncation distance in front of the observation to reduce overwriting effects when the ray passes by close to other obstacles

$$\omega(u) = \begin{cases} e^{-\sigma(u+\epsilon)^2}, & \text{if } u \leq -\epsilon \\ 1, & \text{if } u \leq \tau \, \text{and} \, u > -\epsilon \\ w_{free}, & u > \tau. \end{cases} \quad (6)$$

In our experiments values in the range of $0.1 \leq w_{free} \leq 0.5$ yielded accurate tracking results.

2) Approximate Euclidean Distance Update: The projective distance function is only accurate for observations with viewing angles orthogonal to surface or exactly at the surface.

The less orthogonal the viewing angle becomes the more inaccurate are the update distances. To compensate for the distance biases induced by using the projective distance, we evaluate the use of scan normals to approximate the Euclidean distance (see Figure 3b) to the surface. Therefore, instead of directly using the projective distance, we propose to approximate the distance along a ray from the observation along the scan normal:

$$\boldsymbol{v}_{euclidean}(u) = \boldsymbol{h}_i - u\boldsymbol{n}. \quad (7)$$

Note that $v_{euclidean} \approx v_{projective}$ if the angle of incidence is large which is typically the case for RGB-D cameras as they cannot observe surfaces at oblique angles. However, laser scanners have a high signal to noise ratio which leads to observations even under small angles.

To estimate the scan normal $\boldsymbol{n}$ at the position of the range observation $\boldsymbol{h_i}$ we compute tangents from $\boldsymbol{h_i}$ to all neighboring scan points within a distance d and average the respective normals.

C. Scan Matching

Scan matching is the process of registering laser scans with each other or with an existing map.

For registering scans in TSDF two paradigms are prevalent: ICP based registration [5][6] and direct optimization of the pose on the TSDF [13][7].

We apply a direct optimization of the pose on the TSDF as [13] indicates benefits in accuracy and efficiency. Following their derivations, we phrase scan matching as a Nonlinear Least Squares Problem

$$\underset{\boldsymbol{\xi}}{\operatorname{argmin}} \sum_{i=1}^{N} \left(\Phi_I(\boldsymbol{T}_{\boldsymbol{\xi}}\boldsymbol{h}_i)\right)^2. \quad (8)$$

Φ_I is the bi-linear interpolation of the TSDF grid map M_Φ. We solve the optimization with the Levenberg-Marquardt method using the Ceres Solver [15] and compute gradients with Automatic Differentiation. If odometry is available, we use it for initialization. If no or only inaccurate odometry is available, we apply real-time correlative scan matching [11] to improve the initialization.

V. Global SLAM

Scan matching induces small registration errors, resulting in an accumulation of errors over large distances. Thereby, it yields global inconsistencies in the pose estimate and the map.

To correct these errors, Cartographer generates many small, locally consistent submaps connected in a pose graph. Optimizing the pose graph for its constraints yields a globally consistent map if sufficient constraints have been found. The algorithm uses an efficient branch-and-bound based approach to compute loop closure constraints for occupancy grids. We extend their method to efficiently detect loop closures in TSDFs by adapting the scan matching problem and bounding function for TSDFs.

A. Branch and Bound Scan Matching

The detection of constraints between scans and submaps requires the search for an optimal scan match $\boldsymbol{\xi}^\star$ within a large search window $\mathcal{W}$. If the search window is smaller than the accumulated error in the pose estimate the constraint will be missed. A naive, exhaustive search of the search window is possible. However, for large search windows, it is computationally intractable.

a) Constraint Detection Scan Matching: To detect pose graph constraints, we search an optimal match $\boldsymbol{\xi}^\star$ for a scan $\boldsymbol{H}$

$$\boldsymbol{\xi}^\star = \underset{\boldsymbol{\xi}\in\mathcal{W}}{\operatorname{argmin}} \sum_{k=1}^{K} |\Phi_N(\boldsymbol{T}_{\boldsymbol{\xi}}\boldsymbol{h}_k)| \tag{9}$$

$$s.t. \sum_{k=1}^{K} |\Phi_N(\boldsymbol{T}_{\boldsymbol{\xi}}\boldsymbol{h}_k)| < e_{max} \tag{10}$$

within the search window $\mathcal{W}$. Φ_N is the nearest neighbor interpolation of the TSDF grid map M_Φ. The nearest neighbor interpolation allows an efficient evaluation of the bounding function.

As we are only interested in good matches, we require the solution to have an alignment error smaller than e_{max}. If no solution satisfies this requirement, no constraint is added to the pose-graph.

b) Branch and Bound: Branch and bound approaches represent the solution space as a tree with each node corresponding to a subspace of the solution space. Efficiency in the evaluation of the optimal solution is gained by estimating bounds for the optimal solution within the subtrees. Thereby, the evaluation of large parts of the tree can be omitted and only a small part of the solution space has to be evaluated.

c) Search Window Discretization: To discretize the search window, we choose the angular step size $\delta_r = \arccos(1 - \frac{r^2}{2d_{max}^2})$ and translation step size $\delta_t = r$ with the maximum range d_{max}. Thus, the scan points at the maximum range do not move more than the grid resolution r.

d) Branching rule: The branching rule defines how the solution space of a node is split into the subspaces of its children. A node in the tree is defined by a tuple of integers $c = c_{\{x,y,\phi,h\}} \in \mathbb{Z}^4$ with the node height c_h. Nodes at height c_h combine up to $2^{c_h} \times 2^{c_h}$ translations while representing a single rotation.

A node c at height c_h is branched into up to four nodes at height $c_h - 1$, as we branch once in each translational dimension:

$$\text{branch}(c) = \{c_x, c_x{+}2^{c_h-1}\}\times\{c_y, c_y{+}2^{c_{h-1}}\}\times c_\phi \times c_{h-1}. \tag{11}$$

Child nodes exceeding the search window boundaries are removed.

e) Bounding function: An accurate and efficient evaluation of a lower bound of the optimal solution at an inner node c is crucial to prune large parts of the tree. Approximating the optimal solution within the search window by the optimal match for each scan point within the potential maximum size of the search window yields the bounding function bound(c) as

$$\min_{\boldsymbol{\xi}_c\in\mathcal{W}_c} \sum_{k=1}^{K} |\Phi_N(\boldsymbol{T}_{\boldsymbol{\xi}_c}\boldsymbol{h}_k)| \tag{12}$$

$$\geq \sum_{k=1}^{K} \min_{\boldsymbol{\xi}_c\in\mathcal{W}_c} |\Phi_N(\boldsymbol{T}_{\boldsymbol{\xi}_c}\boldsymbol{h}_k)| = \text{bound}(c). \tag{13}$$

The bounding function can be efficiently precomputed as a grid $\Phi^h_{precomp}$ for each height h

$$\Phi^h_{precomp}([x,y]^T) = \min_{\substack{x'\in[x,x+r(2^h-1)]\\ y'\in[y,y+r(2^h-1)]}} |\Phi_N([x',y']^T)|. \tag{14}$$

The precomputed grid $\Phi^h_{precomp}$ has the same grid structure as $\Phi_{nearest}$, but each cell in the precomputed grid contains the minimum values of the $2^h \times 2^h$ square around it.

With the precomputed grid $M^h_{precomp}$ we can evaluate the bounding function efficiently as

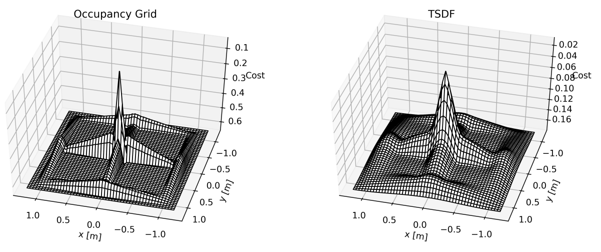
$$\text{bound}(c) = \sum_{k=1}^{K} \Phi^{c_h}_{precomp}(\boldsymbol{T}_{\boldsymbol{\xi}_c}\boldsymbol{h}_k). \tag{15}$$

VI. Results

In this section, we give a detailed evaluation of accuracy and robustness based on an artificial benchmark. We compare the TSDF reconstruction with the projective and the approximate Euclidean updates on simulated data. Furthermore, we compare our method to others in a subset of the Radish dataset. Finally, we demonstrate the capabilities to map a large-scale environments with USAR elements.

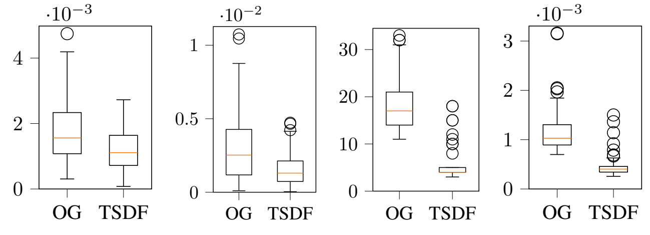
A. Square Benchmark

To compare TSDFs and occupancy grids in a reproducible environment we consider artificial point clouds derived from a square shape with an edge length of $1\,\mathrm{m}$ and a measurement noise with standard deviation $\sigma^2 = 0.01\,\mathrm{m}^2$. The sensor origin is located at the center of the square. We generate an



(a) Occupancy Grid (b) TSDF

Fig. 4: Comparison of the scan matching cost functions for a square with TSDFs and Occupancy Grids. The TSDF provides gradients in a larger area around the optimum, yielding a larger convergence area and higher convergence rates.



(a) Translation Error [m] (b) Rotation Error [rad] (c) Solver Iterations (d) Matching Duration [ms]

Fig. 5: We match 100 simulated scans of a square in a TSDF and an Occupancy Grid (OG). The TSDF scan matcher matches with a higher accuracy while requiring less solver iterations and matching time.

occupancy grid with resolution $r = 0.05\,\mathrm{m}$ and a TSDF with the same resolution and a truncation distance $\tau = 0.25\,\mathrm{m}$.

Fig. 4 shows plots of the scan matching cost functions for translations along the x- and y-axis. The ground truth solution is at $(0, 0)$. The TSDF provides gradients in a larger area around the solution, providing a larger convergence area.

To evaluate the size of the convergence area, we sample the initial position of the scan matching in $2\,\mathrm{cm}$ steps within a window of $1\,\mathrm{m} \times 1\,\mathrm{m}$ around the optimal alignment. The resulting translational errors are shown in Fig. 6. The occupancy grid converges to the optimum within a radius of $0.1\,\mathrm{m}$ whereas the TSDF converges within a radius of $0.35\,\mathrm{m}$.

To evaluate the accuracy and convergence behavior we match 100 point clouds with an initial error of $0.05\,\mathrm{m}$ in uniformly distributed random directions. The resulting errors, solver iterations and matching durations are shown in Fig. 5. The TSDF improves the accuracy significantly by reducing the average translation error by $28.8\,\%$ and the rotational error by $48.5\,\%$ in comparison to the occupancy grid. The solver iterations are reduced by $76.5\,\%$ and the matching time is reduced by $61.1\,\%$.

B. TSDF Update Rules

To evaluate the TSDF update rules we simulate a long, narrow hallway with a square and a circle obstacle inside. The robot is a simulated Pioneer 3DX with a lidar with 180° FOV and a measurement noise with a standard deviation $\sigma^2 = 0.01\,\mathrm{m}^2$. The robot drives a trajectory similar to an eight around the two obstacles. Therefore, the outer ends of the hallways are mainly observed from skewed angles. Fig. 7 shows TSDF maps with projective distance and the approximated Euclidean distance updates.

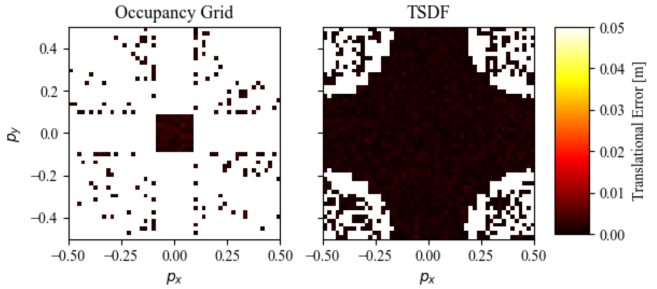
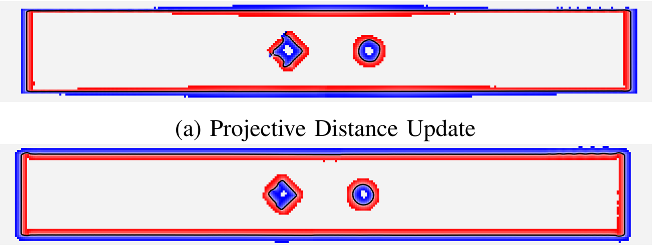


Fig. 6: Translation error after scan matching for initial positions $p = (p_x, p_y)$ within a window of $1\,\mathrm{m} \times 1\,\mathrm{m}$ around the ground truth position. The error is clipped at the grid resolution $r = 0.05\,\mathrm{m}$. The TSDF converges for initial positions within $0.35\,\mathrm{m}$ around the ground truth position whereas the Occupancy Grid converges only for $0.1\,\mathrm{m}$.



(a) Projective Distance Update

(b) Approximated Euclidean Distance Update

Fig. 7: Comparison of TSDF maps with different TSDF update rules in a simulated environment. Red indicates a positive TSDF value, blue encodes negative values. The reconstructed surfaces are indicated by black lines.

The projective approach shows artifacts modeling the square, which is reconstructed more consistently with the Euclidean approximation. In the top right corner, the surface reconstruction with the Euclidean approach shows small jitter.

The projective approach shows a strong overestimation of the absolute distance values close to the corners on the upper and lower edge of the outer rectangle and provides only a narrow truncation band. These errors are to be expected as these cells have only been observed under oblique angles. In contrast, the approximate Euclidean update yields a more consistent estimation of the Euclidean distance within the truncation band around the surface.

C. Radish Benchmark

To compare to other methods, we evaluate our approach on benchmarks from the radish data set [17] using the benchmark measure suggested in [16]. The measure compares the error in relative poses with respect to manually annotated ground truth relations. We choose two data sets with different characteristics, the MIT Killian Court provides more than

TABLE I: Quantitative Error Comparison

	TSDF Projective	TSDF Euclidean	Cartographer	Graph Mapping [16]
MIT Killian Court				
Absolute translational [m]	$\mathbf{0.0276 \pm 0.0232}$	$\mathbf{0.0276 \pm 0.0235}$	0.0324 ± 0.0270	0.050 ± 0.056
Squared translational [m^2]	$\mathbf{0.0013 \pm 0.0089}$	$\mathbf{0.0013 \pm 0.0095}$	0.0018 ± 0.0099	0.006 ± 0.029
Absolute rotational [°]	0.2807 ± 0.2462	$\mathbf{0.2802 \pm 0.2435}$	0.3183 ± 0.2883	0.5 ± 0.5
Squared rotational [deg^2]	0.1394 ± 0.26865	$\mathbf{0.1378 \pm 0.2591}$	0.1844 ± 0.4912	0.9 ± 0.9
Freiburg Bldg. 79				
Absolute translational [m]	$\mathbf{0.0382 \pm 0.0292}$	0.0391 ± 0.0298	0.0395 ± 0.0306	0.056 ± 0.042
Squared translational [m^2]	$\mathbf{0.0023 \pm 0.0044}$	0.0024 ± 0.0045	0.0025 ± 0.0048	0.005 ± 0.011
Absolute rotational [°]	0.4245 ± 0.4610	$\mathbf{0.4204 \pm 0.4606}$	0.4333 ± 0.4735	0.6 ± 0.6
Squared rotational [deg^2]	0.3926 ± 1.2308	$\mathbf{0.3887 \pm 1.1806}$	0.4118 ± 1.2475	0.7 ± 1.7

TABLE II: Runtime Comparison

	TSDF	TSDF	Cartographer
	Projective	Euclidean	
MIT Killian Court			
Wall Clock Time [s]	81.0	74.8	103.5
CPU Time [s]	177.6	162.4	220.8
Memory [MB]	1201.4	1171.2	886.0
Freiburg Bldg. 79			
Wall Clock Time [s]	20.7	18.9	19.1
CPU Time [s]	68.1	107.3	71.7
Memory [MB]	228.6	245.8	215.9

two hours of data with loop closures over large distances and long hallways, whereas the Freiburg Bldg. 79 is a more narrow and cluttered environment with loop closures on smaller scales.

As the data sets differ in sensor configurations and characteristics, we adjust the parameters for each data set individually. For MIT Killian Court we use a grid resolution $r = 0.075\,\mathrm{m}$ and a truncation distance $\tau = 0.15\,\mathrm{m}$. For Freiburg Bldg. 79 we use $r = 0.1\,\mathrm{m}$ and $\tau = 0.15\,\mathrm{m}$. On both data sets, we use odometry information and use the real-time correlative scan matcher with an angular search window of $0.1\,\mathrm{rad}$ to improve the scan matching initialization. As we were able to achieve better results with the current occupancy grid implementation in Cartographer than the results in the original publication, we use the current cartographer implementation performance, as measured by us. For occupancy grids, we choose the same settings for resolution and the real-time correlative scan matcher as for TSDF.

Table I shows the results of our TSDF approach in comparison to the occupancy grids in Cartographer and Graph Mapping [16]. The two TSDF approaches achieve similar accuracy on both benchmarks. In comparison to occupancy grids, the TSDF approaches reduce the absolute translational error by $13.6\,\%$, the squared error by $27.8\,\%$ and the rotational errors by similar margins in the MIT Killian Court data set. In the Freiburg Bldg. 79 data set, the TSDF approaches perform slightly better than the occupancy grids with margins between $1\,\%$ and $5\,\%$.

A potential reason for the small improvement in the Freiburg data set is that it contains many small objects. TSDFs suffer from overwriting effects which degrade the map fidelity when objects smaller than the truncation distance are observed from multiple viewpoints. This is a known issue also described in [6] and requires further investigation.

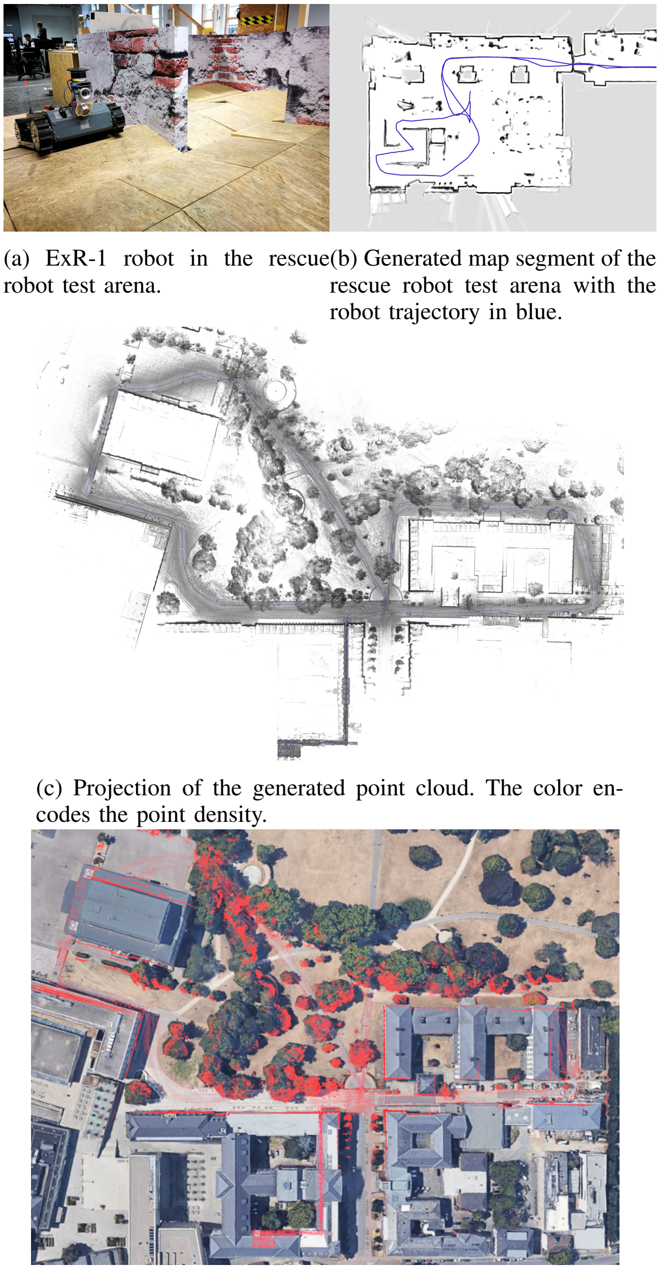
A comparison of the runtimes and memory loads are shown in Table II. For the MIT Killian Court data set the TSDF requires less run time than the occupancy grid. In contrast, Cartographer is slightly faster on the Freiburg Bldg. 79 data set. Overall the TSDF is faster in scan matching, but the map update requires more computations and the bounds for the loop closure branch-and-bound are slightly worse yielding an overall similar runtime. For both data sets the TSDF approach needs $10\,\%$ to $20\,\%$ more memory. An increase in memory is to be expected as TSDF maps store two values, signed distance and weight, per grid cell instead of one for occupancy grids.

D. Darmstadt Campus Experiment

Using data captured at TU Darmstadt Campus with an ExRobotics ExR-1 equipped with a Velodyne VLP-16 lidar, we computed the map shown in Fig. 8. The data set is challenging in multiple aspects. The robot traverses uneven terrain in the rescue robot arena with continuous pitch-and-roll ramps, following the NIST guidelines [18], (see Fig. 8a and Fig. 8b) inducing fast roll and pitch motions. It also covers both narrow indoor hallways and open outdoor spaces in a public park with people walking by. The overall trajectory spans $4160\,\mathrm{s}$ of sensor data and covers approximately $1300\,\mathrm{m}$. The map is computed with a grid resolution $r = 0.05\,\mathrm{m}$, truncation distance $\tau = 0.1\,\mathrm{m}$ and the approximated Euclidean distance update. For larger truncation distances, we observe overwriting effects, when the robot observes walls thinner than the truncation distance from both sides. On a Notebook with an Intel i7-8750H CPU, using up to $3.75\,\mathrm{GB}$ memory the computation finishes after $2364\,\mathrm{s}$ wall time using $7389\,\mathrm{s}$ CPU time, achieving 1.76 times real-time performance.

VII. CONCLUSIONS

In this paper, we presented and evaluated a full 2D SLAM framework based on TSDF maps. To leverage the benefits of TSDFs for lidar-based SLAM, we investigated TSDF update rules and applied an efficient approach to find loop closure constraints in TSDF maps. Our evaluation on publicly available benchmark data showed that the proposed approach achieves higher accuracy in comparison



(a) ExR-1 robot in the rescue robot test arena.

(b) Generated map segment of the rescue robot test arena with the robot trajectory in blue.

(c) Projection of the generated point cloud. The color encodes the point density.

(d) Projection of the generated point cloud in red overlaid over aerial imagery (©Google).

Fig. 8: Darmstadt Campus Experiment

to occupancy grid based methods, while requiring similar runtime. We demonstrated that our approach is able to map a challenging large scale environment with USAR elements in real-time. The proposed method has been published as open source.

As a next step, we are planning to extend the approach to 3D lidar SLAM and further investigate improvements of the normal estimation in sparse scans and the representation of thin surfaces with TSDF.

ACKNOWLEDGMENT

Research presented in this paper has been supported in parts by the German Federal Ministry of Education and Research (BMBF) within the subproject ”Autonomous Assistance Functions for Ground Robots” of the collaborative A-DRZ project (grant no. 13N14861).

The authors gratefully acknowledge the contributions by and fruitful cooperation with all members of the Cartographer Team and Team Hector. Thanks go to Christoph Schütte, Andre Gaschler, Alexander Belyaev, Martin Bokeloh, Sebastian Klose and Damon Kohler.

REFERENCES

[1] W. Hess, D. Kohler, H. Rapp, *et al.*, “Real-time loop closure in 2d lidar slam,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1271–1278.

[2] S. Kohlbrecher, O. Von Stryk, J. Meyer, *et al.*, “A flexible and scalable slam system with full 3d motion estimation,” in *Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on*, IEEE, 2011, pp. 155–160.

[3] G. Grisetti, C. Stachniss, and W. Burgard, “Improved techniques for grid mapping with rao-blackwellized particle filters,” *IEEE transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[4] B. Curless and M. Levoy, “A volumetric method for building complex models from range images,” in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ACM, 1996, pp. 303–312.

[5] R. A. Newcombe, S. Izadi, O. Hilliges, *et al.*, “Kinectfusion: Real-time dense surface mapping and tracking,” in *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, IEEE, 2011, pp. 127–136.

[6] P. Koch, S. May, M. Schmidpeter, *et al.*, “Multi-robot localization and mapping based on signed distance functions,” *Journal of Intelligent & Robotic Systems*, vol. 83, no. 3-4, pp. 409–428, 2016.

[7] J.-D. Fossel, K. Tuyls, and J. Sturm, “2d-sdf-slam: A signed distance function based slam frontend for laser scanners,” in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, IEEE, 2015, pp. 1949–1955.

[8] K. Konolige, G. Grisetti, R. Kümmerle, *et al.*, “Efficient sparse pose adjustment for 2d mapping,” in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, IEEE, 2010, pp. 22–29.

[9] F. Lu and E. Milios, “Globally consistent range scan alignment for environment mapping,” *Autonomous robots*, vol. 4, no. 4, pp. 333–349, 1997.

[10] G. Grisetti, R. Kummerle, C. Stachniss, *et al.*, “A tutorial on graph-based slam,” *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.

[11] E. B. Olson, “Real-time correlative scan matching,” in *2009 IEEE International Conference on Robotics and Automation*, IEEE, 2009, pp. 4387–4393.

[12] D. Droeschel, J. Stückler, and S. Behnke, “Local multi-resolution representation for 6d motion estimation and mapping with a continuously rotating 3d laser scanner,” in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, IEEE, 2014, pp. 5221–5226.

[13] E. Bylow, J. Sturm, C. Kerl, *et al.*, “Real-time camera tracking and 3d reconstruction using signed distance functions.,” in *Robotics: Science and Systems*, vol. 2, 2013.

[14] M. Slavcheva, W. Kehl, N. Navab, *et al.*, “Sdf-2-sdf registration for real-time 3d reconstruction from rgb-d data,” *International Journal of Computer Vision*, pp. 1–22, 2018.

[15] S. Agarwal, K. Mierle, *et al.*, *Ceres solver*, http://ceres-solver.org.

[16] R. Kümmerle, B. Steder, C. Dornhege, *et al.*, “On measuring the accuracy of slam algorithms,” *Autonomous Robots*, vol. 27, no. 4, p. 387, 2009.

[17] A. Howard and N. Roy, *The robotics data set repository (radish)*, 2003. [Online]. Available: `http://radish.sourceforge.net/`.

[18] A. Jacoff, R. Sheh, A.-M. Virts, *et al.*, “Using competitions to advance the development of standard test methods for response robots,” in *Proceedings of the Workshop on Performance Metrics for Intelligent Systems*, ACM, 2012, pp. 182–189.