Robotics and Autonomous Systems 112 (2019) 84–97

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

A pose graph-based localization system for long-term navigation in CAD floor plans

Federico Boniardi [a,*], Tim Caselitz [a], Rainer Kümmerle [b], Wolfram Burgard [a]

[a] University of Freiburg, Georges-Köhler-Allee 80, 79110 Freiburg i. Br., Germany
[b] KUKA, Zugspitzstraße 140, 86165 Augsburg, Germany

HIGHLIGHTS

- Build a pose graph online using trajectory priors to align it to the floor plan.
- Switch to pure localization whenever the environment is sufficiently mapped.
- Estimate the robot pose using relative localization with respect to the pose graph.
- Maintain an up-to-date pose graph in changing environments using Bayes filtering.
- Bound memory consumption and runtime to enable long-term operation.

ARTICLE INFO

Article history:
Received 23 July 2018
Received in revised form 17 October 2018
Accepted 5 November 2018
Available online 10 November 2018

Keywords:
Mobile robotics
Localization
Mapping
SLAM
Adaptive systems

ABSTRACT

Accurate localization is an essential technology for flexible automation. Industrial applications require mobile platforms to be precisely localized in complex environments, often subject to continuous changes and reconfiguration. Most of the approaches use precomputed maps both for localization and for interfacing robots with workers and operators. This results in increased deployment time and costs as mapping experts are required to setup the robotic systems in factory facilities. Moreover, such maps need to be updated whenever significant changes in the environment occur in order to be usable within commanding tools. To overcome those limitations, in this work we present a robust and highly accurate method for long-term LiDAR-based indoor localization that uses CAD-based architectural floor plans. The system leverages a combination of graph-based mapping techniques and Bayes filtering to maintain a sparse and up-to-date globally consistent map that represents the latest configuration of the environment. This map is aligned to the CAD drawing using prior constraints and is exploited for relative localization, thus allowing the robot to estimate its current pose with respect to the global reference frame of the floor plan. Furthermore, the map helps in limiting the disturbances caused by structures and clutter not represented in the drawing. Several long-term experiments in changing real-world environments show that our system outperforms common state-of-the-art localization methods in terms of accuracy and robustness while remaining memory and computationally efficient.

1. Introduction

The ever increasing usage of mobile platforms for industrial and service applications demands navigation systems to combine accurate and robust localization with simplicity of use and deployment. Some of the most advanced localization techniques in industrial settings use the on-board safety LiDAR sensors and rely on maps that require to be built upfront, usually by solving the so called *simultaneous localization and mapping* problem (SLAM). In many circumstances, however, acquiring those maps can be a nuisance since it requires tedious and time-consuming preliminary operations, which increase deployment time and costs for robot manufacturers. In complex environments, expert operators are often needed in order to ensure the consistency of the generated maps and their usability for the robot. Moreover, most common methods use occupancy grid-maps [1] that are now becoming a standard tool for navigation [2] but differ significantly from commonly used drawings of indoor buildings. Grid-maps often present local distortions and pixelation as well as color conventions and encoding that are unusual, making them hardly readable and understandable to inexpert users [3]. Thus, special training is required for shop floor workers and operators in order to get acquainted with such maps, for instance, as visual interfaces for navigation. In

* Corresponding author.
E-mail addresses: boniardi@informatik.uni-freiburg.de (F. Boniardi), caselitz@informatik.uni-freiburg.de (T. Caselitz), rainer.kuemmerle@kuka.com (R. Kümmerle), burgard@informatik.uni-freiburg.de (W. Burgard).

https://doi.org/10.1016/j.robot.2018.11.003
0921-8890/

contrast, architectural drawings are an interesting bridge between an accurate sensor-based robot-centric map representation and an intuitive human-friendly description of indoor environments. They are common in everyday life, for example, on emergency evacuation plans in private and public buildings; they are often available to factory owners and they can be easily manipulated and extended with modern CAD software. Moreover, they represent the unchangeable structures in buildings, thus being an abstract representation of environments that can be flexibly used independently from their actual configuration. Consequently, they can constitute a natural mean for intuitive communication with the navigation tools. Despite these advantages, using such drawings for accurate localization with LiDAR sensors is challenging due to significant discrepancies between the environment observed by the robot and the information represented by the floor plan. The goal of this work is to develop a robust and highly accurate 2D LiDAR-based framework for indoor localization that is designed for long-term operation, even when poorly descriptive CAD floor plans are used.

This paper extends our previous research presented in [4]. In that work, we proposed an approach to estimate the current robot pose in the global reference frame of a prior floor plan. We combined prior measurements obtained by directly matching the LiDAR readings against the floor plan with relative measurements between readings acquired during navigation. Leveraging past observations allows to correct errors and inaccuracies that might occur during navigation due to the absence of information of the architectural drawing. To do so, we used the graph-based mapping framework introduced in [5]. More concretely, we augmented the floor plan with an online built map and used suitable priors for the robot trajectory that constrains the map onto the floor plan. Accordingly, the robot localizes within that map, overcoming the absence of information in the CAD drawing while providing a pose estimation consistent with the floor plan.

Although [4] proved that this approach works robustly in several experimental settings, it still relies on the assumption that the robot is navigating within a fully static environment. This premise poses serious limitations to the usability of the system for continuous operation. The core contribution of this work is to extend the previously proposed system and present a unified framework for long-term localization on prior floor plans, which is able to handle both static and changing environments. For this, we introduce a robust and highly efficient front-end that is capable of maintaining a sparse and up-to-date pose graph that is consistent with the latest observable scenario, without any assumption on the amount of rearrangements that the environment may undergo. We employ a Bayesian filtering approach to track the probability of past LiDAR readings to represent the current scenario. Based on that probability, we constantly discard outdated information from the online built map, resulting in a computationally and memory efficient system usable for long-term operation.

The remainder of this manuscript is organized as follows. Related work is discussed in Section 2. In Section 3 we present a detailed formulation of the problem addressed in this work and the related assumptions. In Section 4 we provide an overview on our LiDAR-based system for floor plan localization and detail the main methodological contributions of the paper. We first outline the system from a general perspective, then focus on describing both the back-end and our robust front-end for changing environments. An extensive experimental evaluation is presented in Section 5. The experiments reported in this paper are original and extend those presented in [4]. Finally, we refer the reader to Table 1 for the mathematical conventions as well as notations adopted throughout the manuscript, unless otherwise specified.

Table 1
The notations adopted in the paper.

Symbol	Description
$\hat{\imath}, \hat{\jmath}$	The natural Cartesian basis of the $\mathbb{R}^2$.
$\mathbb{SE}(2)$	The group of *proper* rigid transformations of $\mathbb{R}^2$.
$\oplus, \ominus$	Compound and inverse operators on $\mathbb{SE}(2)$ [6].
x, y, ϕ	x-, y- and *yaw*-components (*e.g.* as indices).
$\mathbf{p}^{tr}$	Translation component of $\mathbf{p} \in \mathbb{SE}(2)$ (*i.e.* $[\mathbf{p}_x, \mathbf{p}_y]$).
$\mathbf{p}^{R}$	Rotation matrix of angle $\mathbf{p}_\phi$ for $\mathbf{p} \in \mathbb{SE}(2)$.
$\mathbf{p}x$	Rigid transformation of $x \in \mathbb{R}^2$ (*i.e.* $\mathbf{p}^R x + \mathbf{p}^{tr}$).
$[\![\mathbf{p}]\!]$	$\mathbf{p} \in \mathbb{SE}(2)$ as an element of $\mathbb{R}^2 \times (-\pi, \pi] \subset \mathbb{R}^3$.
Σ, Ω	Covariance and related information matrix.
$\|x\|$	Euclidean norm: $(x^\top x)^{1/2}$.
$\|x\|_\Sigma$	Mahalanobis norm: $(x^\top \Omega x)^{1/2}$.
$x_{t_0:t_n}$	The sequence $(x_{t_0}, \ldots, x_{t_n})$, $t_0 < t_1 < \ldots < t_n$.
$x_{t_0:t_n} \setminus x_{t_k}$	Removal of x_{t_k} from $x_{t_0:t_n}$.
$\#X$	Number of elements in a finite set.
$\#x_{t_0:t_n}$	Number of element in a finite sequence.
$f(x;\theta)$	A function f with arguments x and parameters θ.
Δ_t	Sequence of relative measurements at time t.
Π_t	Sequence of prior measurements at time t.

2. Related work

Mobile robot localization is a vast research field that has been widely investigated. Its crucial role for autonomous robot navigation made it one of the major topics in the robotics community over the last decades, resulting in mature techniques that deliver robust and efficient solutions. Many methods that have been successfully developed and deployed treat the localization problem within the Bayesian state estimation framework. The earliest approaches used Kalman and histogram filter-based techniques [7,8]. More recently, particle filter-based approaches, called Monte Carlo Localization methods (MCL) [9] proved to be one of the most efficient and versatile. Over the years, these algorithms were extended in different variants to account for movable and unmapped obstacles [10,11]. Typically, in order to represent the static parts of the environments, they use maps computed with the same sensor modality as the one used for localization. The most common representation are so called occupancy grid-maps, introduced by Elfes [1], which describe the space in terms of probability of grid-cells of being occupied.

Inspired by the graph-based SLAM and multi-robot communities, other authors investigated localization approaches that relaxed the assumption of using maps obtained by fusing sensor measurements, even forgoing the need of global consistency. They proposed the concept of *relative localization*. The map is usually encoded as a set of relative pose constraints between nodes (pose graph) where each node is associated with a measurement against which the robot localizes, for instance, by using scan-matching algorithms. The robot pose is therefore defined only with respect to the currently matched node or set of nodes and not with respect to a global reference frame. In the context of LiDAR-based localization, this was extensively used by Sprunk et al. [12], who defined maps as user-taught trajectories, and further generalized to more complex pose graphs by Mazuran et al. [13]. Noticeably, Schiotka et al. [14] proposed a sensor model for MCL using those *scan-based* maps to overcome the need of memory expensive occupancy grid-maps. In the context of graph-based mapping, scan-based maps seem to be firstly investigated by [15].

Architectural floor plans as aid for robot localization have received scarce attention in research and few works have investigated methods to create precise and robust localization systems for such maps. In most cases, their usage seems to be motivated by the infeasibility to create proper maps using the sensor on board the robots or the devices used. Siddiqi et al. [16] presented an extension of MCL using Wi-Fi signals to estimate its pose on CAD floor plans. Ito et al. [17] proposed an MCL-based sensor

fusion approach to track the pose of a mobile device on CAD drawings. The system uses an RGB-D camera and IMU for visual-inertial odometry, while the depth information is exploited to create a set of beams that are used as sensor readings. The particle initialization is improved employing the Wi-Fi signal, which allows faster convergence to a unimodal distribution of the particle set. A similar idea was proposed by Winterhalter et al. [18], that presented a localization system for a Google Tango tablet based on MCL on CAD floor plans of indoor environments. The proposal distribution of MCL is computed using visual-inertial odometry and, again, the depth data of the RGB-D camera on board the device is compared with a 3D model of the environment obtained by elevating the walls encoded in the floor plan. Hile et al. [19] developed a vision-based localization system on a CAD drawing by triangulating landmarks extracted from the image and matched onto the floor plan. Luo et al. [20] presented a method that uses an architectural floor plan of a building and extracts both metrical and topological information. More precisely, they store both the floor plan and the room numbers and use a camera to detect and interpret the room plates as well as an ultrasonic sensor to match doorways and significant landmarks. The measurements are then fused using a Bayesian filter. In the same spirit, another MCL-based approach called SeDAR was presented by Mendez et al. [21]. The authors developed a localization system for a mobile robot that uses a floor plan and an RGB-D camera. Their core concept is to precompute a likelihood field not only for the beam endpoints obtained by the depth sensor, but also for the semantics of the environment. They compute likelihood fields for doors or windows which are detected by the robot using the RGB-D image. A more general likelihood field model is obtained as a mixture of the single fields and used for the sensor model of MCL.

All above mentioned methods assume either a fully static environment or the possibility to ignore its dynamic parts. They rely on a static world that is exhaustively represented in the map and use robust models that aim at reducing the disturbing effect of measurements that cannot be explained by the static part of the environment. An approach for localization in semi-static environments was presented in Meyer-Delius et al. [22]. The authors proposed an extension of MCL that uses a sensor model based on the joint likelihood of a static map and local maps built during operation, called *temporary maps*. A mapping process is triggered whenever the current LiDAR scan does not match the static map or the temporary map in which the robot is operating. If the map does not sufficiently explain the current scan, the system discards it and restarts a temporary mapping process. An approach that accounts for changing environments during mapping was presented in [23]. In that work the authors proposed a generalization of grid-maps that encode state transitions for every grid-cell to model changeable environments. Such dynamic grid-maps have been coupled with MCL techniques by Tipaldi et al. [24] for lifelong localization. The authors used an MCL-based approach to track the robot pose and the current state of the map according to its transition properties in order to localize a mobile robot robustly within a heavily changing environment. Other methods leverage temporal information about the evolution of map features. Kraijník et al. [25] proposed an approach based on Fast Fourier Transforms that learns repetitive patterns of changes in the environment. The learned model is used to predict occupancy values of the map cells. Additionally, a model for short-term persistence of observations is employed. In the same spirit, the pose graph-based SLAM community has investigated methods to encode dynamics and provide similar generalizations. Biber et al. [26] proposed a mapping system that uses different timescales to represent the changes in the environment. Measurements are replaced or removed using an online learned model. Another method, called Dynamic Pose Graph, was proposed by Walcott-Bryant et al. [27]. The authors introduced an extended version of 2D LiDAR scan-based maps that encode additional information such as the state of a node being active or inactive. The transition of nodes between active and inactive is determined according to the number of active angular bins in the related LiDAR scans. An abstract approach for modeling changing environments was introduced by Rosen et al. [28]. In this work, the authors presented a Bayesian filter based on the survival analysis framework to model whether features detected during mapping have to be considered out-of-date. They proposed a recursive update that incorporates the results of feature detection and time-dependent priors on the lifetime of features.

The approach presented in this paper borrows ideas and methods from this background. It uses methods for graph-based SLAM with priors to leverage past observations and online build a scan-based map of the environment that is globally consistent with the CAD floor plan. This has multiple advantages. First, in contrast to [24], it benefits from storing the history of readings and trajectory: the online estimated map can thus be corrected during navigation, which is a well known advantage of pose graph-based SLAM compared to particle filter-based mapping. Second, the method presented in [24] suffers from a high memory consumption when the map resolution is increased, particularly if a large number of particles is used to achieve accurate tracking, while our proposed method does not depend on the resolution of the floor plan. Third, our method uses the global consistency of the online built map to disambiguate parts of the environment that are perceptually indistinguishable from the floor plan, for instance when flat furniture or equipment is covering an entire wall. This is a clear advantage in terms of robustness compared to methods that only rely on robust sensor models to match the current measurements against the floor plan. In turn, the availability of a prior floor plan is beneficial for the SLAM system, in particular to maintain an up-to-date pose graph, which is a crucial aspect of our localization system. Our method does not need to store the history of past configurations to localize and update the map as in [26]: this results in a highly memory and computationally efficient system. Furthermore, in contrast to [27] (and arguably to [26]) our approach for change detection and map updates does not rely on the assumption that the amount of change still allows correct scan registration. In comparison to [22], our method presents significant enhancements. First, we use LiDAR readings for online mapping irrespectively of how the current scan matches the prior map. The online map is therefore already partially built during initial detouring towards areas where the floor plan poorly represents the environment, which avoids unnecessary drift. Second, we leverage the global consistency of the built map to correct possibly incorrect registrations against the floor plan. As discussed in [24], whenever the system presented in [22] commits to wrong alignments of temporary maps, there is little chance of recovering. Third, instead of relying on temporary maps to cope with changing environments, we employ a more fine-grained approach that uses a locality criterion to choose the set of LiDAR scans to query for change detection and selects scans individually for removal. This prevents the system to discard parts of the online built map even if they can later be exploited for localization. Finally, our approach does not require any knowledge about the temporal evolution of the environment [28] and we do not assume prior temporal models such as periodic changes [25]. Conversely, we use a Markov-based decay model that relies on time-independent priors on the likelihood of changes.

3. Problem statement

In this work we address the problem of pose tracking using a known CAD-based floor plan that misses significant information about the environment. Formally, given an approximate initial

guess $\mathbf{x}_0 \in \mathbb{SE}(2)$ for the robot pose and a CAD drawing encoded as a binary image $\mathscr{I} \in \{\text{F}, \text{O}\}^{W \times H}$ with a certain resolution, our goal is to estimate the robot pose $\mathbf{x}_t \in \mathbb{SE}(2)$ at any time with respect to a global reference frame $\mathscr{F}_f$ of the floor plan, even within an environment that might be subject to changes and rearrangements. We assume the robot to employ a 2D LiDAR sensor. We henceforth consider every scan S to be a sequence of 2D Cartesian points $(s_i)_i$ expressed in the reference frame of the sensor, which we assume to coincide with the one of the robot. We note that we are not interested in solving the global localization problem, that is, to estimate the robot pose without any initial knowledge on the initial pose of the robot with respect to $\mathscr{F}_f$. From the application perspective, this is often not required as in many industrial and service applications the starting pose for the operation is usually a docking station or charging spot whose location is known upfront by the factory floor operators or customers.

Similarly to [24] and [27], throughout this work we assume that the environment presents low dynamic objects and features, that is, despite changes that might occur in the environment, at any time t all readings recorded in the LiDAR measurement S_t are obtained from objects with zero velocity with respect to the global reference frame $\mathscr{F}_f$. As pointed out by Walcott et al. [27], this assumption does not affect the generality of the approaches since several algorithms for detection and tracking of highly dynamic objects have been proposed in literature and successfully deployed in real-world applications, *e.g.* [29,30]. To further support this assumption, we also remark that the building block of the method proposed in this paper, namely 2D LiDAR-based *Iterative Closest Point* (ICP), has been proven to be highly robust and reliable even when people are walking nearby the robot as well as when small objects move within the field of view of the sensor [31].

In general, we do not assume that (a part of) the floor plan is always visible during navigation. This assumption poses serious limitations to methods that exclusively rely on directly matching the current observation against the floor plan (*e.g.* MCL), even for robust variants that can cope with unmapped obstacles and dynamics. Operating in areas where the floor plan is not observable causes drift to accumulate, eventually resulting in localization failures.

In contrast to [27,32], we do not implicitly assume that, over a longer period of time, scan-matching is sufficient to solve the data association problem intrinsic in a SLAM system and thus loop closing to be always possible when revisiting previous locations over a longer period of time. Instead, we allow changes in the environment that are not limited to small features and isolated parts of rooms and therefore cannot be handled by an outlier rejection method. More concretely, we assume that rearrangements in the environment can significantly reshape the landscape observed by the robot during navigation as that is a common situation that frequently occurs in real-world settings. As discussed in [24], this assumption makes the localization task harder since the maintenance of an up-to-date map becomes crucial for localization accuracy, which might be dramatically affected by wrong data associations. Since our goal is to localize with respect to a prior CAD floor plan, we are not interested in storing the history of previous configurations of the environment or generating a map that mirrors it in every detail. This constitutes a substantial difference between our approach and the method proposed in [27]. In particular, key for us are the following aspects:

- High memory and computational efficiency so that the system not only remains computationally tractable as in [27], but also solves the mapping problem at least in *near real-time*.
- In contrast to accurate and exhaustive mapping, building a map that stores the past configurations is of lower priority to us than having a map that encodes just enough information to enable accurate localization. For instance, displacements of small objects in empty rooms should not trigger any instance of change detection or map update (as shown in [27]) since they would unlikely result in scan-matching failures. In fact, such circumstances can be easily treated by either outlier rejection or association trimming during scan registration.

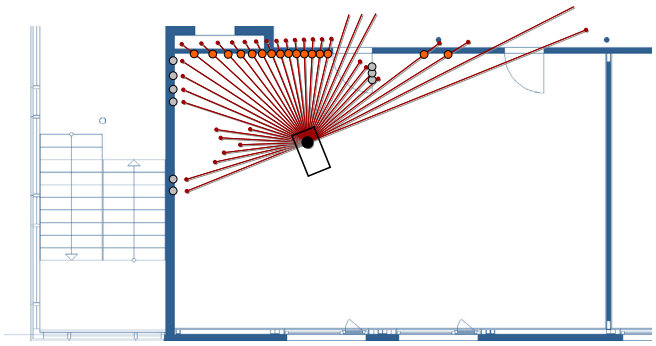


Fig. 1. A pictorial example of an association set used in the GICP-based registration of a LiDAR scan against a CAD floor plan. Associations obtained using the distance map are reported in gray, those obtained via ray-casting in orange. Whenever the distance map associates a pixel on the wrong side of the wall, the association candidate is replaced with a pixel obtained by ray-casting.

In light of these goals and differently from [28], we avoid discarding parts of the map according to a time-dependent decay. Such a policy would result in removing all nodes that exceed the expected lifetime, unless fresh observations are obtained. As a consequence, the system could not leverage past observations for localization, which is particularly beneficial in areas where the environment represented in the floor plan is not observable.

Finally, we do not encode any information about "passes" [27], that is, we assume that changes might occur at any time during operation and not necessarily during different navigation phases, for instance, after the map has been fully acquired. Furthermore, the robot does not need to start each navigation task from the latest pose in the previous localization run. We only assume that the very first run should start in a location where the visible environment sufficiently matches the floor plan to ensure that an initial pose alignment is possible just by registering the LiDAR reading onto the CAD drawing.

4. Localization system

As mentioned in the previous sections, our goal is to estimate the current robot pose $\mathbf{x}_t \in \mathbb{SE}(2)$ in the reference frame of the floor plan, while a globally consistent scan-based map of the environment is built online. This map aims at overcoming the absence of details in the CAD drawing by creating constraints between the map scans, thus correcting the errors produced by inaccurate registration of LiDAR measurements against the floor plan. Similarly to [13–15], the scan-based map $\mathscr{M}_t \triangleq \langle \mathbf{x}_{0:t}, S_{0:t}, \mathscr{F}_f \rangle$ estimated by the robot consists of a set of trajectory poses $\mathbf{x}_{0:t} \in \mathbb{SE}(2)^{n+1}$ in the reference frame $\mathscr{F}_f$ of the floor plan and a collection of LiDAR measurements $S_{0:t}$ obtained at those poses. Following the graph-based SLAM formulation [33] with prior information [5], such a map can be inferred by estimating the trajectory $\widehat{\mathbf{x}}_{0:t}$ that maximizes the *posterior* probability of a sequence of relative measurements $\Delta_t \triangleq (\mathbf{z}_{t_i,t_j})_{ij} \in \mathbb{SE}(2)^m$ between pairs of trajectory poses visited during navigation:

$$\widehat{\mathbf{x}}_{0:t} \in \arg\max_{\mathbf{x}_{0:t}} p(\mathbf{z}_{t_i,t_j}, \ldots, \mathbf{z}_{t_r,t_s} \mid \mathbf{x}_{0:t}) p(\mathbf{x}_{0:t}), \tag{1}$$

that is, by maximizing the *likelihood* of the relative measurements weighed by a *trajectory prior*. In the proposed system, all relative measurements are obtained by scan-matching between selected pairs of poses. Henceforth, consistently with the conventional naming used in the context of graph-based SLAM, we will synonymously use *scan-based map* and *pose graph*, *trajectory poses* and *nodes* as well as *measurements* and *constraints*.

Assuming the trajectory poses $\mathbf{x}_{0:t}$ to be pairwise independent, the relative measurements $(\mathbf{z}_{t_i,t_j})_{ij}$ to be conditionally independent with respect to $\mathbf{x}_{0:t}$ as well as all probability terms to be normally distributed, Eq. (1) can be expressed as *log-posterior* and formulated as the *nonlinear least-squares* optimization problem

$$\widehat{\mathbf{x}}_{0:t} \triangleq \arg\min_{\mathbf{x}_{0:t}} \sum_{t_i,t_j} \chi^2(\mathbf{x}_{t_i}, \mathbf{x}_{t_j}; \mathbf{z}_{t_i,t_j}) + \sum_{t_k} \chi^2(\mathbf{x}_{t_k}; \mathbf{z}_{t_k}), \tag{2}$$

where $\chi(\mathbf{x}_{t_i}, \mathbf{x}_{t_j}; \mathbf{z}_{t_i,t_j}) \triangleq \|[\![\mathbf{z}_{t_i,t_j}]\!] - [\![\ominus \mathbf{x}_{t_i} \oplus \mathbf{x}_{t_j}]\!]\|_{\Sigma_{t_i,t_j}}$ is the relative error term for two poses with respect to the associated relative measurement and $\chi(\mathbf{x}_{t_k}; \mathbf{z}_{t_k}) \triangleq \|[\![\mathbf{z}_{t_k}]\!] - [\![\mathbf{x}_{t_k}]\!]\|_{\Sigma_{t_k}}$ is the error term induced by a prior measurement $\mathbf{z}_{t_k} \in \mathbb{SE}(2)$ for the trajectory pose $\mathbf{x}_{t_k}$. In the previous definitions, Σ_{t_i,t_j} and Σ_{t_k} are the covariance terms of the underlying normal distributions of the relative and prior constraints respectively. We henceforth refer to Π_t as the sequence of prior measurements $\mathbf{z}_{0:t} \in \mathbb{SE}(2)^{n+1}$.

In practice, the estimation above is often obtained by solving a generalized version of the problem, which assumes that all error terms have distributions of the form $\eta e^{-\kappa(\chi)}$, where κ is a non-negative real-valued function, typically called *kernel*. The resulting problem becomes

$$\widehat{\mathbf{x}}_{0:t} = \arg\min_{\mathbf{x}_{0:t}} \sum_{t_i,t_j} \kappa_{\mathrm{r}}\left(\chi(\mathbf{x}_{t_i}, \mathbf{x}_{t_j}; \mathbf{z}_{t_i,t_j})\right) + \sum_{t_k} \kappa_{\mathrm{p}}\left(\chi(\mathbf{x}_{t_k}; \mathbf{z}_{t_k})\right), \tag{3}$$

where the kernel terms $\kappa_{\mathrm{r}}, \kappa_{\mathrm{p}}$ are usually *quasiconvex* functions that encode super-Gaussian distributions for the error terms in Eq. (2). Robust kernels make the optimization problem less sensitive to wrong measurements, commonly referred to as *outliers*. The estimation in Eq. (3) can be efficiently solved with standard optimization techniques such as the Levenberg–Marquardt algorithm.

In the above framework, the relative measurements can therefore be used to generate a globally consistent map of the environment at every time t while the trajectory priors anchor it to the floor plan. The nodes $\widehat{\mathbf{x}}_{0:t}$, and accordingly also the current pose $\mathbf{x}_t$, are softly constrained to be aligned with the reference frame $\mathscr{F}_f$ of the CAD drawing. Thus, relative localization with respect to the stored nodes $\widehat{\mathbf{x}}_{0:t}$ results in a pose estimation that is consistent with the CAD floor plan. Note that CAD floor plans can have moderate metrical inaccuracies which might result in slight inconsistencies between prior and relative constraints. These inaccuracies can be averaged during the trajectory estimation by scaling the covariance terms of the relative measurements. A natural choice is to set $\Sigma'_{t,t_j} \triangleq n_t \Sigma_{t,t_j}$ ($\Omega'_{t,t_j} \triangleq \frac{1}{n_t}\Omega_{t,t_j}$), where $n_t \in \mathbb{N}_{>0}$ is the number of relative measurements related to the current pose $\mathbf{x}_t$. In the least-square optimization in Eq. (2), n_t balances of error terms related to relative measurements with those related to prior constraints.

4.1. Floor plan trajectory prior

In order to generate priors $\mathbf{z}_{0:t}$ for the robot trajectory, we adapted the Generalized-ICP framework (GICP) proposed by Segal et al. [34] for scan-to-floor plan matching. The approach is akin to that proposed in [35].

As mentioned in Section 3, we assume the floor plan to be given as a 2D image $\mathscr{I}$ with a certain resolution. In order to compute the prior $\mathbf{z}_t$, at every update time t we register the LiDAR

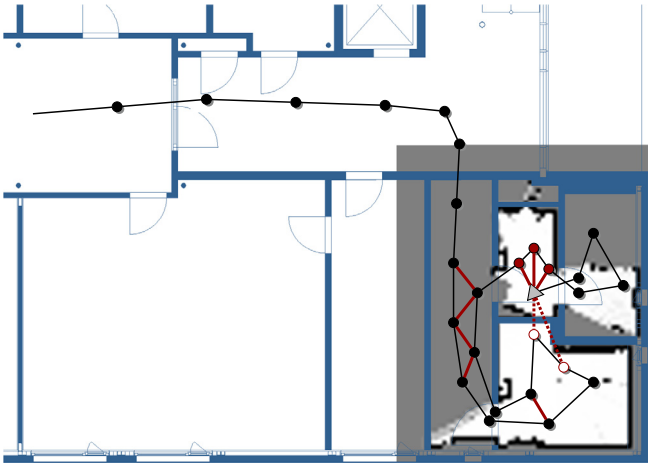


Fig. 2. Topological check for the loop closure candidates $\mathscr{C}_t$. The local grid map $\mathscr{O}_{\mathscr{C}_t}$ is overlaid onto the floor plan. The gray triangle represents the current robot pose $\mathbf{x}_t$. Trajectory poses are shown in black, accepted candidates in red, and the discarded candidates in white. Nodes whose connecting segments do not lie in the free space of the local grid map are discarded from the loop closures set (dashed edges).

measurements S_t to $\mathscr{I}$ using an approach similar to ICP. Namely, we iteratively compute an association set $\mathscr{A} \triangleq \{\langle s_i, m_i\rangle\}_i$ between the LiDAR endpoints and the occupied pixels in $\mathscr{I}$ and estimate the transformation that best aligns the associations by solving the following nonlinear optimization problem:

$$\mathbf{z}_t \triangleq \arg\min_{\mathbf{z}\in\mathbb{SE}(2)} \sum_{\langle s,m\rangle\in\mathscr{A}} \kappa\left(\|\mathbf{z}s - m\|_{\Sigma(\mathbf{z};s,m)}\right), \tag{4}$$

with $\Sigma(\mathbf{z}; s, m)$ being the two-dimensional variant of the GICP covariance matrix

$$\Sigma(\mathbf{z}; s, m) \triangleq \left[\mathbf{z}^R R_s\right] \begin{bmatrix} \nu & 0 \\ 0 & 1 \end{bmatrix} \left[\mathbf{z}^R R_s\right]^\top + R_m \begin{bmatrix} \eta & 0 \\ 0 & 1 \end{bmatrix} R_m^\top, \tag{5}$$

where R_s and R_m are the 2D rotation matrices that align $\hat{\imath}$ to the normal of the scan at s and the normal of the floor plan at m respectively. Again κ is a robust kernel that limits the effect of wrong associations and $\nu, \eta > 0$ are covariance terms that weigh each association along the related normals [34]. For the sake of the well-posedness, we always consider the image normal to point from occupied to free pixels and the scan normals to be directed towards the origin of the sensor.

The GICP error function in Eq. (4) accounts for the local geometry surrounding the associations. In particular, it downweighs those with mismatching normals. This is particularly suitable for the application discussed in this work since CAD floor plans typically only report the essential features of buildings, such as walls, but not specific objects or even clutter in the environment and wrong associations are therefore frequent. However, walls and clutter typically have different geometrical structures and thus the influence of such wrong associations is substantially reduced in the optimization.

Following [33], we can use the information matrix Ω_t of the system obtained by linearizing the error terms in Eq. (4) after the last iteration to compute the covariance matrix Σ_t for the prior measurement $\mathbf{z}_t$. Concretely, the information matrix Ω_t can be estimated as

$$\sum_{\langle s,m\rangle\in\mathscr{A}} \left[\left.\frac{\partial f(\mathbf{z}_t \oplus \mathbf{v}; s, m)}{\partial \mathbf{v}}\right|_{\mathbf{v}=\mathbf{0}}\right]^\top \left[\left.\frac{\partial f(\mathbf{z}_t \oplus \mathbf{v}; s, m)}{\partial \mathbf{v}}\right|_{\mathbf{v}=\mathbf{0}}\right], \tag{6}$$

where

$$f(\mathbf{z}; s, m) \triangleq r\left(\Lambda(\mathbf{z}; s, m)^\top (\mathbf{z}s - m)\right), \tag{7}$$

with $r(x) \triangleq \|x\|^{-1} x \sqrt{\kappa(\|x\|)}$ and $\Lambda(\mathbf{z}; s, m)$ is the lower triangular Cholesky factor of $\Omega(\mathbf{z}; s, m)$.

Despite the GICP-based back-end, a robust policy for selecting associations is still crucial for an accurate registration of the LiDAR measurements against the floor plan. Due to the image-based encoding of the floor plan, the GICP error function is ineffective or even not well-defined in cases that commonly occur if a vanilla closest-point-based association policy is used, namely:

- Endpoints are associated with occupied pixels in parts of the image with zero image gradient, thus the normals are not defined. This happens, for instance, to pixels representing thick walls.
- Endpoints are associated with the wrong side of a wall, sometimes referred to as *see-through-walls effect* [36]. As a consequence, the resulting normal alignment is correct up to 180° and does not cause any downweighing by $\Sigma(\mathbf{z}; s, m)$ defined in Eq. (5).

To overcome these issues without overly depleting of the association set, we use the following policy to select candidate pixels. At every iteration, we require that every association $\langle s, m \rangle$ must satisfy the following three conditions:

(a) $\|\mathbf{z}_{\text{iter}} s - m\| \leq \delta_{\text{iter}}$,
(b) The image normal $\hat{n}_m$ of m exists.
(c) $\hat{n}_m^\top \mathbf{z}_{\text{iter}}^R \hat{n}_s > 0$, where $\hat{n}_s$ is the scan normal of s.

Condition (c) enforces the relative angle between the floor plan normal $\hat{n}_m$ and the transformed scan normal $\mathbf{z}_{\text{iter}}^R \hat{n}_s$ to not exceed 90°. For every endpoint s, we first select the closest occupied pixel to $\mathbf{z}_{\text{iter}} s$ as candidate pixel on $\mathscr{I}$. If this pixel satisfies the above conditions, the association is added to the association set, otherwise another candidate is selected by ray-tracing along the beam direction and the same acceptance criteria are applied. An example of an association set is shown in Fig. 1.

Observe that, while the first candidate can be computed in constant time by caching the *distance transform* of the image $\mathscr{I}$, ray-tracing might result in a computational overhead that depends on the amount of free space in the floor plan and the selected resolution. However, the distance transform can be leveraged to dramatically speed-up ray-casting by setting the distance step used to search along the ray equal to the value of the distance transform at the currently visited pixel, often referred to as *sphere-tracing* [37,38]. Owing to the definition of distance transform, the number of steps is substantially reduced while rays are prevented to overshoot obstacles and the ray-tracing algorithm becomes independent from the floor plan resolution. Therefore, the total computational overhead introduced by the robust association policy is negligible with respect to the optimization step.

4.2. Long-term mapping

During long-term operation, the proposed system must prevent the scan-based map $\mathscr{M}_t$ and the collection of relative measurements $(\mathbf{z}_{t_i,t_j})_{ij}$ to grow indefinitely, thus exhausting memory and computational resources. Several methods for pose graph compression and sparsification have been proposed in literature and provide offline approaches to reduce the number of nodes and constraints without compromising the quality of the resulting map [39–42]. Conversely, we prevent the number of nodes and relative constraints to increase by using an approach which neither requires any post-processing nor additional computational cost.

More specifically, following common approaches for loop closing in two-dimensional SLAM [43,44], at every time step t a set of loop closure candidate nodes $\mathscr{C}_t$ is selected by using a threshold distance $\rho_t > 0$ with respect to the current robot pose and excluding the most lately added nodes. In addition, a local *occupancy grid map* $\mathscr{O}_{\mathscr{C}_t}$ is generated from the LiDAR measurements associated with the nodes in $\mathscr{C}_t$ using the *log-odd* approach described in [36]. The local map is used to discard candidates that are not topologically consistent with the local environment surrounding the robot. We do this by ray-tracing along the line segments connecting the current robot pose and the candidates (see Fig. 2). As further checks, we validate candidate nodes by evaluating the amount of overlapping field of view via re-projection as well as by estimating the cross-visibility of LiDAR beams as discussed in [45]. Whenever the number of resulting valid loop closures exceeds a threshold value N_{loc}, we estimate the current robot pose by using the local *maximum a posteriori* optimization

$$\begin{aligned} \widehat{\mathbf{x}}_{t_0:t} &\triangleq \arg\min_{\mathbf{x}_t} \sum_{t_j} \kappa_{\text{r}} \left(\chi(\mathbf{x}_t, \mathbf{x}_{t_j}; \mathbf{z}_{t,t_j}) \right) + \kappa_{\text{p}} \left(\chi(\mathbf{x}_t; \mathbf{z}_t) \right), \\ &\in \arg\max_{\mathbf{x}_t} p(\mathbf{z}_{t,t_p}, \ldots, \mathbf{z}_{t,t_q} \mid \mathbf{x}_t) p(\mathbf{x}_t) \end{aligned} \tag{8}$$

without performing the full trajectory estimation in Eq. (2) and without storing the latest LiDAR measurement and trajectory pose in $\mathscr{M}_t$ as well as relative and prior measurements in Δ_t and Π_t respectively. Note that this approach might result in under-mapped areas for only two specific configurations of the pose graph which can be easily treated as special cases:

- At time t_*, the vehicle drives into a newly visited area which is bounded by the range ρ_t $(t > t_*)$ and remains in that area for a certain time. The local optimization in Eq. (8) might be triggered before relative measurements have corrected the poses $\mathbf{x}_{t_*:\tau}$ $(\tau \leq t)$. However, such a situation can be easily detected as the robot would try to localize against a set of candidates that are solely connected by incremental measurements.
- The robot enters an unmapped part of the environment after a local estimation was performed. In this case, the candidate set $\mathscr{C}_t$ might be empty due to a doorway or narrow passage that causes all candidates to be discarded by the topological validation. Consequently, the resulting pose graph would be disconnected. Although the prior measurement $\mathbf{z}_t$ prevents the estimated pose $\widehat{\mathbf{x}}_t$ to be decoupled from the floor plan in principle, in case of severe occlusions the resulting wrong prior measurements would not be countered by correct relative constraints. Again, such a case can be easily detected by observing that there are no relative constraints connected to the current node $\mathbf{x}_t$. The connectivity can then be restored by adding the latest trajectory pose $\widehat{\mathbf{x}}_{t-1}$ including the related LiDAR measurement S_{t-1} to the map $\mathscr{M}_t$ and storing the relative constraints associated with the latest candidates $\mathscr{C}_{t-1}$ together with the prior constraint in Δ_t and Π_t respectively.

Algorithm 1 Connectivity-preserving map pruning.

```
procedure MapPruning(M; p*)
    x̂_t, x̂_t̄                          // earliest and latest pose in the trajectory
    for x̂_τ in x̂_{t:t̄} do
        if odds[x̂_τ] < odd(p*) then
            T ← DFT(M_t; x̂_τ)                    // depth-first search tree
            if #Children(x̂_τ; T) = 1 then          // not an articulation
                S_{t:t̄} ← S_{t:t̄} \ s_τ
                Δ_t̄ ← Δ_t̄ \ {⟨z_{τ,t_j}, Σ_{τ,t_j}⟩ | t ≤ t_j ≤ t̄}
                Π_t̄ ← Π_t̄ \ {(z_τ, Σ_τ)}
```

4.3. Changing environments

Long-term operation often involves substantial changes in the environment. In order to keep the robot localized in such scenarios, it is crucial to maintain an up-to-date map. Significant changes

in the environment dramatically affect the data association for loop closures and can cause wrong relative measurements, which eventually compromise localization accuracy. Even if the selection method for loop closure candidates described in Section 4.2 correctly filters scans that are not usable for loop closing, a mapping process would be triggered, causing the number of nodes and LiDAR scans in the map $\mathscr{M}_t$ to increase, which results in a substantial computational overhead. In order to overcome these problems, we propose an efficient method to detect significant changes and maintain only nodes that store LiDAR measurements that are consistent with the latest observations obtained by the robot. The method does not rely on the assumption that loops can be closed consistently by matching the current scan against scans stored in $\mathscr{M}_t$. We leverage the validity of ICP-based scan-matching to estimate the likelihood whether a LiDAR scan related to a trajectory pose is consistent with the currently observable environment or not. We then use this likelihood to prune the past nodes and to robustify the trajectory estimation.

Algorithm 2 Localization in prior floor plans.

procedure LOCALIZE(S_t, $\mathbf{u}_t$; d, α, ρ, h, N_{loc}, p^*, $p^{\text{O}|\text{A}}$, γ)
  $\widehat{\mathbf{x}}_{\underline{t}}, \widehat{\mathbf{x}}_{\bar{t}}$ // *earliest and latest pose in the trajectory*
  $\mathscr{D}_t \leftarrow \emptyset$ // *the set of current relative measurements*

  $\langle \mathbf{z}_{t-1,t}, \Sigma_{t-1,t} \rangle \leftarrow$ ICP($S_{t-1}, S_t, \mathbf{u}_t$) // *incremental motion*
  if $\|\mathbf{z}^{tr}_{t,t-1}\| < d \wedge |\mathbf{z}^{\phi}_{t,t-1}| < \alpha$ **then**
    skip

  $\mathbf{x}_t \leftarrow \widehat{\mathbf{x}}_{t-1} \oplus \mathbf{z}_{t-1,t}$
  $\langle \mathbf{z}_t, \Sigma_t \rangle \leftarrow$ GICP($S_t, \mathbf{x}_t$; $\mathscr{I}$) // *floor plan prior*

  $\mathscr{C}_t \leftarrow \emptyset$
  for $\mathbf{x}_\tau \in \widehat{\mathbf{x}}_{\underline{t}:\bar{t}}$ **do** // *find loop-closures candidates*
    if $\|\mathbf{x}^{tr}_t - \mathbf{x}^{tr}_\tau\| < \rho \wedge |t - \tau| > h$ **then**
      $\mathscr{C}_t \leftarrow \mathscr{C}_t \cup \{\tau\}$

  $\mathscr{O}_t \leftarrow$ OCCUPANCYGRIDMAP($\mathscr{C}_t$)

  for $\tau \in \mathscr{C}_t$ **do** // *filter loop closure candidates*
    if odds $[\widehat{\mathbf{x}}_\tau] <$ odd (p^*) **then**
      $\mathscr{C}_t \leftarrow \mathscr{C}_t \setminus \{\tau\}$
    if FIELDOFVIEWOVERLAP($\mathbf{x}_t, S_t, \mathbf{x}_\tau, S_\tau$) $< \beta$ **then**
      $\mathscr{C}_t \leftarrow \mathscr{C}_t \setminus \{\tau\}$
    if $\neg$LINECANCONNECT($\mathbf{x}_t, \mathbf{x}_\tau$; $\mathscr{O}_t$) **then**
      $\mathscr{C}_t \leftarrow \mathscr{C}_t \setminus \{\tau\}$

  for $\tau \in \mathscr{C}_t$ **do** // *compute loop-closures and beliefs*
    if $\langle \mathbf{z}_{t,\tau}, \Sigma_{t,\tau}, \varepsilon \rangle \leftarrow$ ICP($S_t, S_\tau, \ominus \mathbf{x}_t \oplus \mathbf{x}_\tau$) **then**
      odds $[\widehat{\mathbf{x}}_\tau] \leftarrow \phi(\varepsilon; \theta)$
      if odds $[\widehat{\mathbf{x}}_\tau] >$ odd (p^*) **then**
        $\mathscr{D}_t \leftarrow \mathscr{D}_t \cup \{\langle \mathbf{z}_{t,\tau}, \Sigma_{t,\tau} \rangle\}$
    else
      odds $[\widehat{\mathbf{x}}_\tau] \leftarrow \frac{\gamma}{1-p^{\text{O}|\text{A}}}$ odds $[\widehat{\mathbf{x}}_\tau]$ + odd $\left(p^{\text{O}|\text{A}}\right)$

  if $\bar{t} = t - 1$ **then** // *add incremental measurement*
    $\mathscr{D}_t \leftarrow \mathscr{D}_t \cup \{\langle \mathbf{z}_{t-1,t}, \Sigma_{t-1,t} \rangle\}$

  if $\#\mathscr{D}_t < N_{\text{loc}}$ **then** // *full trajectory optimisation*
    $\mathbf{x}_{\underline{t}:t} \leftarrow (\widehat{\mathbf{x}}_{\underline{t}:\bar{t}}, \mathbf{x}_t)$
    $S_{\underline{t}:t} \leftarrow (S_{\underline{t}:\bar{t}}, S_t)$
    $\Delta_t \leftarrow \Delta_{\bar{t}} \cup \mathscr{D}_t$
    $\Pi_t \leftarrow \Pi_{\bar{t}} \cup \{\langle \mathbf{z}_t, \Sigma_t \rangle\}$
    MAPPRUNING($\mathscr{M}_t$; p^*)
    $\widehat{\mathbf{x}}_{\underline{t}:t} \leftarrow$ OPTIMIZE($\Delta_t, \Pi_t, \mathbf{x}_{\underline{t}:t}$)
  else // *pose optimisation*
    MAPPRUNING($\mathscr{M}_t$; p^*)
    $\widehat{\mathbf{x}}_t \leftarrow$ OPTIMIZE($\mathscr{D}_t, \langle \mathbf{z}_t, \Sigma_t \rangle, \mathbf{x}_t$)

Concretely, similarly to *e.g.* [23,25,28], we introduce binary time-dependent random variables $H^t_{t_i} \in \{\text{A}, \text{O}\}$ associated with the trajectory poses $\mathbf{x}_{t_i}$ and LiDAR measurement S_{t_i} that encode whether, at time t, S_{t_i} matches the latest robot observation S_t. For the sake of clarity, in such a case we will say that the node $\mathbf{x}_{t_i}$ is *actual* (A) or *old* (O) otherwise. Additionally, we define binary random variables $V^t_{t_i} \in \{\top, \bot\}$ that encode whether ICP was able to estimate a relative measurement $\mathbf{z}_{t,t_i}$ during loop-closing ($\top$) or S_t and S_{t_i} are not comparable ($\bot$), for instance, due to an insufficient number of associations. We define the belief of $H^t_{t_i}$ as $\text{bel}(H^t_{t_i}) \triangleq p(H^t_{t_i} \mid V^{1:t}_{t_i})$. The observability of $H^t_{t_i}$ depends upon $V^t_{t_i}$ since it is only possible to compare the current observation S_t with the past observation S_{t_i} if they could be validly registered. In this case, the belief of $H^t_{t_i}$ can be obtained by scoring the average squared misalignment error ε between associations after the last iteration of ICP. That is, if $V^t_{t_i} = \top$, the belief simplifies to

$$\begin{aligned} \text{bel}(H^t_{t_i} = \text{A}) &= p(H^t_{t_i} = \text{A} \mid V^t_{t_i} = \top) \\ &\triangleq \phi(\varepsilon; \theta), \end{aligned} \tag{9}$$

where $\phi : \mathbb{R}_{\geq 0} \to [0, 1]$ is a monotonically decreasing function with shape parameters θ and such that $\phi(0) = 1$. The first equality in Eq. (9) holds since $H^t_{t_i}$ is independent from $V^{1:t-1}_{t_i}$ whenever ICP can estimate a relative measurement at time t. On the contrary, if $V^t_{t_i} = \bot$, the variable $H^t_{t_i}$ cannot be observed directly. However, applying Bayes rule as in [23], the following filter equation holds:

$$\text{bel}(H^t_{t_i}) = \eta p(V^t_{t_i} = \bot \mid H^t_{t_i}) \sum_{H^{t-1}_{t_i}} p(H^t_{t_i} \mid H^{t-1}_{t_i}) \text{bel}(H^{t-1}_{t_i}), \tag{10}$$

where the summation is intended to be over the binary values assumed by $H^{t-1}_{t_i}$. Setting $p^t_{t_i}(\text{X}|\text{Y}) \triangleq p(H^t_{t_i} = \text{X} \mid H^{t-1}_{t_i} = \text{Y})$ for $\text{X}, \text{Y} \in \{\text{A}, \text{O}\}$, under the assumption that a node cannot return *actual* after becoming *old*, that is $p^t_{t_i}(\text{A} \mid \text{O}) = 0$, we can express Eq. (10) in terms of *odds* as the following non-homogeneous recursive update:

$$\begin{aligned} o^t_{t_i} &= \frac{\gamma_{t_i}}{p^{\text{A}|\text{A}}_{t_i}} o^{t-1}_{t_i} + \text{odd}\left(p^t_{t_i}(\text{O}|\text{A})\right) \\ &= \frac{\gamma_{t_i}}{1 - p^t_{t_i}(\text{O}|\text{A})} o^{t-1}_{t_i} + \text{odd}\left(p^t_{t_i}(\text{O}|\text{A})\right), \end{aligned} \tag{11}$$

with $\gamma_{t_i} \triangleq \frac{p(V^t_{t_i} = \bot | H^t_{t_i} = \text{O})}{p(V^t_{t_i} = \bot | H^t_{t_i} = \text{A})}$ and with $o^t_{t_i} \triangleq \text{odd}\left(\text{bel}(H^t_{t_i} = \text{O})\right)$ and $\text{odd}(p) \triangleq (1 - p)^{-1} p$. Accordingly, the evolution of the belief of a node only depends on prior probabilities $p^t_{t_i}(\text{O}|\text{A})$ of nodes to become outdated and the ratio γ_{t_i} of the probabilities of scan-matching failures given the actuality of nodes.

Since we assume that a node can only degrade to an *old* node, as discussed in [28], we can use the estimated belief $\text{bel}(H^t_{t_i})$ to further prune the map $\mathscr{M}_t$ by removing nodes that are considered outdated, that is, whenever the belief $\text{bel}(H^t_{t_i} = \text{A})$ drops below a tolerance value. Although this might sometimes result in discarding some nodes that are falsely considered outdated, it does not adversely affect the localization system since the map is newly enriched with the most recent measurements. Note that preserving the connectivity of the graph is crucial to remove the gauge freedom during the full-trajectory optimization when the robot accesses areas in which priors measurements cannot be obtained. In order to keep the pose graph connected at any time, we selectively avoid to remove the nodes that would result in disconnecting the graph, so called *articulation points*, as described in Algorithm 1. Checking whether a trajectory pose is an articulation point for the map $\mathscr{M}_t$ can be done efficiently with linear complexity $O(\#\Delta_t + \#\mathbf{x}_{\underline{t}:t})$ [46], where $\mathbf{x}_{\underline{t}}$ is the earliest node added to the map, that is, with a negligible runtime overhead. Note that even though articulations points are preserved, they are not used for

loop closing once they are considered to be outdated. This pruning method bounds the size of the map and the amount of relative measurements $\mathbf{z}_{t,\tau}$ that need to be computed at every update step, thus limiting the computational and memory requirements of the system. The loop closure candidates resulting after map pruning are then used in the localization update (see Algorithm 2).

In order to incorporate the information of $\text{bel}(H_{t_i}^t)$ into the estimations of Eq. (2) and Eq. (8), we employ *Dynamic Covariance Scaling* (DCS) [47]. DCS models the presence of unreliable relative measurements by means of a specific kernel κ_{dcs}. Essentially, DCS translates the *Switchable Constraints* method (SC) [48] into the robust kernel-based optimization framework. In SC the error terms $\chi(\mathbf{x}_{t_i}, \mathbf{x}_{t_j}; \mathbf{z}_{t_i,t_j})$ are weighed by coefficients $s_{t_i,t_j} \in [0, 1]$ that toggle the related constraints on or off. The best values $\widehat{s}_{t_i,t_j}$ together with the optimal trajectory $\widehat{\mathbf{x}}_{0:t}$ are obtained by optimizing the resulting joint error function

$$h_{\text{sc}}(s, \mathbf{x}_{0:t}) \triangleq \sum_{i,j} s_{t_i,t_j}^2 \chi^2(\mathbf{x}_{t_i}, \mathbf{x}_{t_j}; \mathbf{z}_{t_i,t_j}) + \sum_{i,j} \omega_{t_i,t_j}(1 - s_{t_i,t_j})^2 + \sum_k \kappa_{\text{p}}\left(\chi(\mathbf{x}_{t_k}; \mathbf{z}_{t_k})\right), \tag{12}$$

where the *switching priors* $\omega_{t_i,t_j} \geq 0$ are weights that can be interpreted as a prior confidence that the constraints $\mathbf{z}_{t_i,t_j}$ are not outliers. As shown in [47], the error function in Eq. (12) is equivalent to

$$h_{\text{dcs}}(\mathbf{x}_{0:t}) \triangleq \sum_{i,j} \kappa_{\text{dcs}}\left(\chi(\mathbf{x}_{t_i}, \mathbf{x}_{t_j}; \mathbf{z}_{t_i,t_j})\right) + \sum_k \kappa_{\text{p}}\left(\chi(\mathbf{x}_{t_k}; \mathbf{z}_{t_k})\right), \tag{13}$$

where the DCS kernel is defined as $\kappa_{\text{dcs}}(\chi; \omega) \triangleq \sigma(\omega)^2\chi^2$, with the adaptive scaling factor $\sigma(\omega) \triangleq \min\{1, (\omega + \chi^2)^{-1}\omega\}$ for the switching prior ω.

We leverage the estimated beliefs of nodes to be up-to-date by setting $\omega_{t,t_j} \triangleq \text{bel}(H_{t_j}^t = \text{A})$ as parameters for the DCS kernel $\kappa_{\text{r}} \triangleq \kappa_{\text{dcs}}$ in Eqs. (2) and (8). This is correct if loop closure outliers are only caused by the presence of outdated nodes. It is reasonable to assume this for the proposed system as it substantially reduces other sources for wrong data association in loop closing

- by localizing on a floor plan that prevents the robot to drift even if no relative measurements are available, thus reducing the need of a large search range for loop closure candidates,
- by filtering the candidates using the topological check and further comparing the related LiDAR measurements as described in Section 4.2.

5. Experimental evaluation

We tested our system in several real-world scenarios in order to assess its robustness, accuracy as well as the runtime and memory requirements. We compared our method with MCL by using CAD drawings as binary occupancy grid-maps. All data was collected by teleoperating a Pioneer 3-DX® differential drive robot and a KUKA omniRob® omnidirectional drive robot, both equipped with a 30 m range SICK S300 Professional® laser rangefinder on board with 270° of field of view and 541 beams. All computations have been performed on an 8-core 4.00 GHz Intel® Core™ i7-4790K CPU. In all experiments, we drove the Pioneer 3-DX® with constant linear and angular velocity of approximately 0.5 m/s and 0.6 rad/s, while the KUKA omniRob® was commanded at approximately 1.0 m/s and 1.0 rad/s. A summary of the experimental settings as well as the quantitative analysis are reported in Table 2.

The implementation of our system relies on the g^2o framework [49] for the scan-based map implementation as well as the optimization in Eqs. (3), (8), and (4), which are all solved via Levenberg–Marquardt method. We use the *point-to-line* ICP

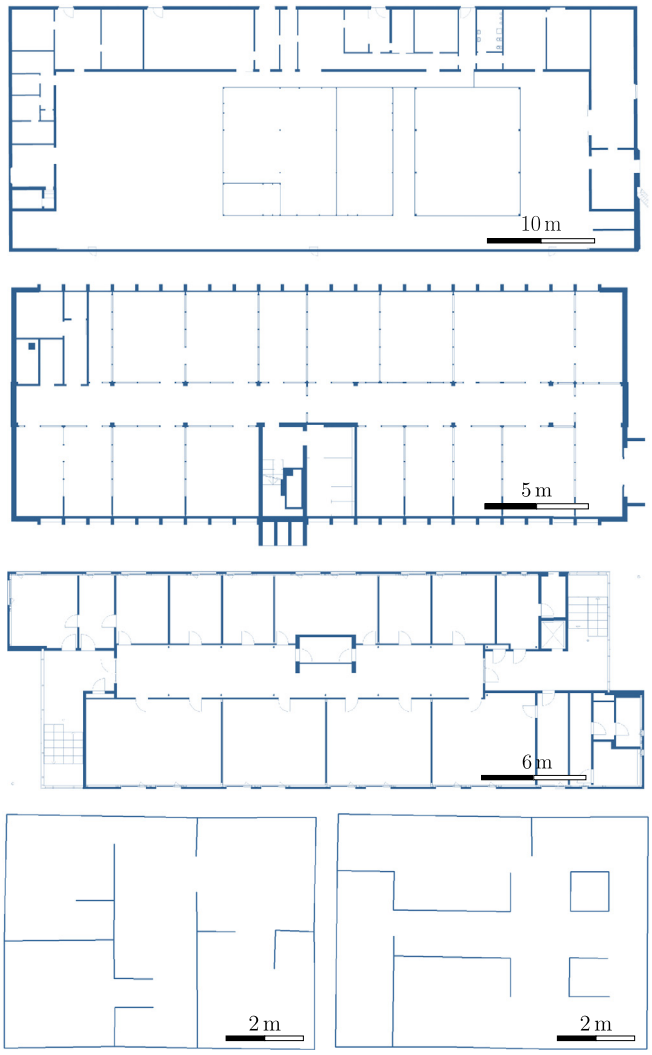


Fig. 3. The CAD drawings used in the experiments with scales. From top to bottom: *Building078* (*Fr003*, *Fr078*), *Building079* (*Fr079*), *Building080* (*Fr080*), *Fr001* (bottom-left) and *Fr002* (bottom-right).

implementation provided by the *C(anonical) Scan Matcher* [50,51] to compute the relative measurements $\langle \mathbf{z}_{t_i,t_j}, \Sigma_{t_i,t_j} \rangle$ as well as the misalignment error ε in Eq. (9). For all experiments we used a common set of parameters that were manually selected. In particular, referring to Algorithm 2, we set $d = 0.75$ m, $\alpha = 0.5$ rad, $\rho = 2$ m, $h = 5$, $N_{\text{loc}} = 5$, $p^* = 50\,\%$, and $p^{\text{o|A}} = 15\,\%$. For the likelihood estimation in Eq. (9), we used $\phi(\varepsilon; \theta) \triangleq \mathcal{N}(\max\{\varepsilon, \mu^*\}; \mu^*, \sigma^*)$, where $\mathcal{N}(x; \mu^*, \sigma^*)$ is a Gaussian probability density function and $\mu^* \approx 20$ mm is the 95th percentile of the empirical distribution function of the average misalignment errors $(\varepsilon_k)_k$ measured in a separate dataset of a static environment. The standard deviation term $\sigma^* = 100$ mm has been chosen to be conservative. For all experiments, the CAD images were exported with a resolution of 10 px/mm and we computed the normal fields using a 3×3 Scharr convolution [52]. To compute the trajectory prior we used Huber kernels, $\delta_{\text{iter}} = \frac{15\,\text{mm}}{\sqrt{\text{iter}}}$, $\eta = 0.05$ and $\nu = 0.05$.

5.1. Datasets

In order to evaluate our system we recorded six datasets with the aim of reproducing the circumstances and challenges that occur during a continuous long-term operation. Each dataset is composed of multiple chunks (*days*) that have been recorded during

Fig. 4. In the top row the configuration of *Fr078-day-4*. In the third and fourth rows, the scan-based map built online for dataset *Fr078* at the end of each day. Different days are reported from top-left (day-1) to bottom right (day-4). The gray/red scale encodes the ratio between older and newly added scans: in red only LiDAR scans obtained during the current day, in light gray only the scans inherited from previous days. Observe that in day-2 a new room was explored (top-right part of the CAD drawing) and the navigation covered all the accessible areas (same as in day-1). The unchanged areas are correctly detected. Conversely, In day-3 and day-4 only the wide hall was used for the navigation. Note the car have slightly moved. In all cases, the pose graphs have been correctly updated.

different days. The scenarios were subject to changes ranging from minimal displacements of objects to complete rearrangements, while within each day no substantial changes occurred. The robot was provided with an initial pose estimate for each day, which did generally not coincide with the final pose of the previous day. However, at the initial location it was always possible to register the current scan against the latest scan-based map built during the previous day. This poses no loss of generality to the assumptions described in Section 3. The floor plans used in the experiments are reported in Fig. 3. The datasets are grouped as follows:

- *Ground-truth datasets*: Three datasets (*Fr001*, *Fr002*, and *Fr003*) composed of multiple days recorded in the robot hall of the University of Freiburg (*Building078*). For each day, the scenario was set up by placing and rearranging panels and objects. During teleoperation we tracked the ground-truth pose of the robot using an external motion capture system with ten Raptor-E cameras. *Fr001* and *Fr002* were obtained by concatenating the chunks used for the accuracy evaluation presented in [4]. In order to obtain floor plans for these two datasets, we put reflective markers on the panels and measured their positions with the same motion capture system. To avoid biases, the cluttered scenario was in one case prepended to the uncluttered one and appended in the other case. *Fr003* uses the floor plan of the building.
- *Long-term datasets*: Three long datasets (*Fr078*, *Fr079*, and *Fr080*) recorded in three different buildings of the University of Freiburg (*Building078*, *Building079*, and *Building080*). Since the official CAD drawings present metrical inaccuracies, we could not consistently compare the poses estimated by our localization system with any ground-truth. Consequently, we mainly used these datasets to assess the long-term performance of our system in terms of computational and memory efficiency as well as its qualitative robustness. Nonetheless, in order to provide some quantitative analysis of the accuracy, we estimated an approximate ground-truth trajectory as follows:

  1. We used a graph-based SLAM system to obtain an accurate trajectory and map.
  2. We manually overlapped this SLAM map with the CAD drawing and horizontally/vertically stretched /compressed the map using constant scaling factors to compensate for the inconsistencies. More precisely, given a point $m \in \mathbb{R}^2$ on the SLAM map (*e.g.* the translation components of a trajectory pose or a LiDAR endpoint), we fit the following linear model

$$\begin{bmatrix} m'_x \\ m'_y \end{bmatrix} \triangleq \begin{bmatrix} p_x \\ p_y \end{bmatrix} + \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} (\mathbf{t}\, m)_x - p_x \\ (\mathbf{t}\, m)_y - p_y \end{bmatrix}, \tag{14}$$

  where $\mathbf{t} \in \mathbb{SE}(2)$ is the transformation that aligns the SLAM map to the CAD drawing, $s_x, s_y > 0$ are correction scales, and p_x, p_y are fixed pivots representing the coordinates of some reference landmarks on the CAD floor plan that can be clearly matched to the SLAM map, such as the end of a corridor or a clearly distinguishable wall.

Table 2
A summary of the experimental settings and results of the quantitative evaluation. Errors are reported as linear and angular deviation from the ground-truth. MCL reports the best results with respect of the runs as well as the sensor model as discussed in Section 5.5. The days marked with an asterisk are the dataset used for the evaluation proposed in [4].

Dataset	Robot	Ground-tr.	Chunk	Duration [min]	Distance [m]	Area [m^2]	RMSE [mm | °]				Runtime [ms]	
							Ours		MCL		Ours	MCL
Fr001	Pioneer 3-DX®	Mocap	day-1*	10	129	50	15	0.62	25	1.64	31 ± 8	23 ± 6
			day-2*	11	142		23	1.87	31	2.61	35 ± 12	24 ± 5
Fr002	Pioneer 3-DX®	Mocap	day-1*	10	159	50	52	3.43	55	3.73	30 ± 10	28 ± 9
			day-2*	11	166		26	0.82	27	1.53	38 ± 11	28 ± 8
Fr003	KUKA omniRob®	Mocap	day-1	10	294	60	52	0.96	310	2.28	15 ± 8	46 ± 6
			day-2	10	365		48	1.17	275	3.20	21 ± 11	46 ± 6
			day-3	10	388		51	1.31	285	4.46	20 ± 13	46 ± 6
			day-4	11	365		65	0.98	698	10.77	21 ± 8	46 ± 6
Fr078	Pioneer 3-DX®	Manual	day-1	52	1501	1900	209	1.22	562	1.55	29 ± 15	54 ± 18
			day-2	65	1687		163	1.16	502	2.09	50 ± 18	55 ± 18
			day-3	32	902		166	1.20	693	2.75	43 ± 16	56 ± 14
			day-4	22	567		127	1.44	521	3.25	54 ± 21	50 ± 14
Fr079	Pioneer 3-DX®	Manual	day-1*	34	804	570	95	1.34	113	1.41	24 ± 14	10 ± 2
Fr080	Pioneer 3-DX®	Manual	day-1*	60	1405	724	87	1.38	98	1.23	37 ± 26	8 ± 2
			day-2	59	1473		63	1.43	65	1.57	53 ± 30	8 ± 2

3. We transformed the SLAM trajectory to be consistent with the CAD drawing by applying the same scaling transformation. The resulting trajectory is considered the ground-truth trajectory on the floor plan.

We linearly interpolated the ground-truth poses to obtain a continuous ground-truth trajectory. In order to reduce the error introduced by interpolation, the SLAM trajectory was computed updating the SLAM system with high frequency, namely whenever relative cumulative motion exceeded 100 mm or 10°. Only the CAD drawing for *Building080* is metrically accurate ($s_x, s_y \approx 1$), while the SLAM maps of *Building078* and *Building079* required a compression of 1.7% ($s_x = 0.983$) and 2.3% ($s_x = 0.977$) of its longitudinal dimension respectively, which corresponds to a mismatch of 825 mm and 834 mm. In all cases the transversal inconsistencies are negligible ($s_y \approx 1 \pm 10^{-5}$).

In accordance with the discussion in Section 3, none of the datasets contain highly dynamic objects except for few people occasionally walking within the field of view of the sensor. However, all algorithms used in this work are robust to a limited amount of outliers and spurious readings. Thus, the made assumptions are not violated since the effect of people walking around the robot is negligible. This also applies to MCL [24,36], which we use as benchmark algorithm.

5.2. Robustness

Even in challenging scenarios, where the environment represented in the floor plan is hardly distinguishable from occluding objects, such as large furniture covering a wall, the proposed system was able to cope with the inconsistencies caused by inaccurate prior measurements. This is achieved by leveraging the relative constraints to build a globally consistent map that resolves ambiguities in data associations during scan-to-floor plan registration and, consequently, improves the trajectory prior estimation over time. In certain situations observed in *Fr078* and *Fr079*, the robot failed to properly register the LiDAR readings against walls almost fully occluded by cupboards. As the robot revisited these areas, the correct alignment was recovered.

Substantial changes in the environment pose another challenge as wrong data associations cause inaccurate loop closure measurements. Our system is not substantially affected as we explicitly handle this situation. In *Fr080* the reconfiguration consists of displacement of furniture and introduction of clutter while the visibility of walls is mainly preserved. Even though artificially created, *Fr001*, *Fr002* and *Fr003* reproduce a similar scenario. In these datasets, the robot remained robustly localized and no qualitative effects were visible during operation. In *Fr078*, areas are totally rearranged and new configurations do not maintain any similarity. The system successfully detected the changes and remapped the areas to reliably localize the robot (see Fig. 4). Fig. 5 shows the map for *Fr078-day-4* obtained by the localization system we proposed in [4], which assumes a static environment. The inconsistency of the maps is caused by incorrect loop closure measurements obtained by registering LiDAR scans recorded in different environment configurations. This shows that loop closing using scan-matching is not always feasible, which is consistent with the assumptions in Section 3. In contrast to [4], the map obtained by the proposed system remains globally consistent over time as can be seen in Fig. 4. The robustness of the proposed method is also demonstrated in *Fr080-day-2*, where our method provides accurate pose estimates while [4] suffers from major failures caused by incorrect scan-matching due to significant changes. The system presented in [4] delivered a linear RMSE of 171 mm compared to 63 mm (see Table 2) and a maximum linear error of 1868 mm compared to 461 mm of the proposed system.

5.3. Accuracy

According to the results of the experiments reported in Table 2, the system was consistently able to provide accurate pose estimates. The average RMSE over all ground-truth datasets is (41 ± 18) mm in position and (1.39 ± 0.90)° in orientation. Consistently with the results reported in [4], in both *Fr001* and *Fr002*, the localization accuracy was higher in the uncluttered scenarios due to the presence of objects occluding the panels represented by the floor plan. Such circumstances lead to wrong data associations during registration against the floor plan since parts of the objects are hardly distinguishable from the panels. Furthermore, our system achieves a performance comparable to [4], namely 15 mm|0.62° vs. 12 mm|0.52° in *Fr001-day-1* and 52 mm|3.43° vs. 42 mm|2.90° in *Fr002-day-1*, which shows that the proposed generalization to changing environments does not substantially affect its accuracy in static environments. In *Fr003*, the accuracy slightly dropped with respect to *Fr001* and *Fr002*. We attribute this to three major reasons that make *Fr003* a more challenging dataset. First, minor or even no parts of the floor plan are visible during navigation. Second, the environment is extensively rearranged between different days.

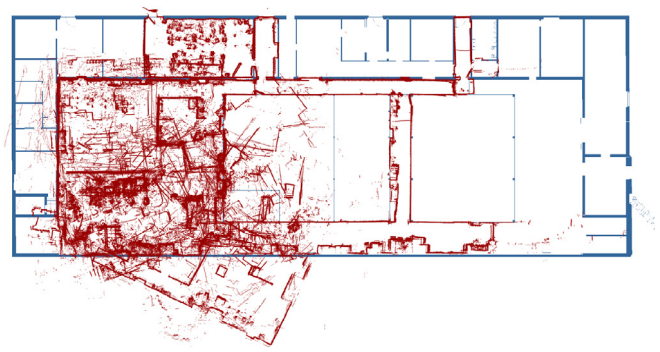


Fig. 5. The maps at the end of *Fr078-day-4* with the system proposed in [4], that is, without pose graph pruning and adapting the kernel parameters based on bel(H_t^L). The reconfiguration of the environment results in accumulating wrong relative constraints that eventually produce severe failures of the system.

Third, the sensor on board the robot moves at a significantly higher velocity (higher angular speed, larger distance from the center of rotation).

Table 2 reports substantially higher localization errors for the long-term datasets compared to the ground-truth datasets, except for *Fr080*. As discussed in Section 5.1, only the CAD drawing of *Building080* is metrically accurate, thus, it allows results that are comparable to those of the ground-truth datasets. Conversely, in *Fr078* and *Fr079*, the metrical inconsistency of the floor plans with the real-world led to errors in estimating the trajectory prior, which in turn caused the lower accuracy. Registering a scan against an inconsistent floor plan can produce inaccurate priors, as a proper rigid transformation may not be sufficient to achieve a consistent alignment even if perfect data associations are given. The average RMSE over all long-term datasets is (130 ± 52) mm and $(1.31 \pm 0.11)^\circ$ in position and orientation respectively.

5.4. Runtime and memory requirements

The experiments reported in Fig. 6 show that the system has a bounded runtime, in particular there is no substantial increase even over longer periods of time. For a localization update, our system required an average of (26 ± 12) ms on the ground-truth datasets and (41 ± 12) ms on the long-term datasets. The highest peak of approximately 157 ms occurred during *Fr080-day-2*. Observe that the average runtime is always lower for the first days since the full pose graph optimization is computationally more expensive on the following days as the map contains substantially more nodes on average. This effect is visible in Fig. 6. In order to reduce the dependence of the computational efficiency on the size of the map, in the experiments the full trajectory optimization in Eq. (1) is restricted to a local submap , which we selected as the *closed kth neighborhood* of the node $\mathbf{x}_t$ in $\langle \mathscr{M}_t, \Delta_t \rangle$ $(k = 20)$. In all experiments, the system ran on average in real-time with respect to the frequency of the sensor on-board the robot (12 Hz $\approx$ 83 ms).

As expected, the system runtime is highly correlated to the number of trajectory poses, which defines the memory requirements of the system. As clearly shown in Fig. 7, the amount of poses is bounded by the mapping method described in Section 4.2, in particular compared to a standard system without pose graph sparsification. The number of poses remained approximately stationary after sufficient exploration. In *Fr079* new areas were discovered throughout the day. In general, given a dataset, the number of nodes mainly depends on the perception range ρ used for loop closures and the minimum number of loop closures required to run a local optimization N_{loc} (see Algorithm 2). The choice of these parameters results in a trade-off between accuracy (small ρ and large N_{loc}) and computational/memory requirements (large ρ and small N_{loc}).

5.5. Comparison with Monte Carlo localization

We compare our method against two different implementations of MCL that are robust to unmapped obstacles and low dynamics. For our tests we used a *beam-based model* (MCL-BBM) [36] and a *likelihood field model* with distance saturation (MCL-SLF) [13] as measurement model $p(S_t \mid \mathbf{x}_t)$. Applying kernels on particle weighing increases localization robustness against unmapped obstacles as they prevent particles from being excessively downweighed by LiDAR readings obtained from obstacles not represented in the map. To further improve the MCL baseline, we corrected the odometry-based proposal distribution $p(\mathbf{x}_t \mid \mathbf{u}_t, \mathbf{x}_{t-1})$ by incremental scan-matching. We executed multiple localization runs for every dataset using both sensor models. Table 2 reports the results with the highest linear accuracy without specifying which model performed better as it is irrelevant in the scope of this evaluation. Henceforth MCL will denote the better performing model. Since it was not possible to find a common parameter set that achieved consistent performance in all datasets, we tuned the most significant parameters for each dataset. We kept the number of particles constant and used the same distance thresholds for filter updates as for our method. We used 5000 particles for MCL-SLF and 2500 particles for MCL-BBM due to its higher computational cost. We chose a large number of particles to enable MCL to recover from localization failures and to reduce the dependence of the experimental results on the randomicity in the algorithm. We tuned the saturation distance and the noise magnitude for MCL-SLF, the mixing coefficients for MCL-BBM and the noise parameters for the proposal distribution. MCL-BBM used sphere-tracing to improve the runtime performance.

The experiments show that our method is comparable with MCL whenever the floor plan is at least partially observable during operation. Table 2 reports higher accuracy of the proposed system for the ground-truth datasets *Fr001* and *Fr002*, however, the improvement is below the resolution of the floor plan and only comparable performance can be claimed. This is confirmed by the results on the long-term datasets *Fr079* and *Fr080*, where clutter rarely covered significant parts of the environment represented by the floor plans. In contrast, our method significantly outperformed MCL on *Fr003* since it contains situations where the LiDAR measurements only capture large structures and movable panels that are not represented by the floor plan. This is a frequent situation in real-world scenarios, for instance in industrial applications. Here, our method achieved a substantial improvement in accuracy, namely an average of (338 ± 197) mm and $(4.07 \pm 3.88)^\circ$. An improvement in accuracy is also confirmed by the results of *Fr078*. On this dataset MCL-SLF performed substantially worse that MCL-BBM, failing in 75 % of the days with a linear error reaching up to more than 6 m. Both required a dramatic reduction in the noise terms of the sample distribution in order to prevent wrong data associations, compromising the capability of the system to promptly recover from inaccurate pose estimates. In *Fr078* our method outperformed MCL on average by (403 ± 85) mm and $(1.15 \pm 0.7)^\circ$. Overall, the experiments show that the proposed system has a similar runtime, is more robust, and is substantially less sensitive to the choice of parameters compared to MCL.

6. Conclusions

In this work we presented a system that uses CAD floor plans for robust and accurate long-term localization. The proposed method employs a graph-based SLAM approach that uses priors from architectural drawings to generate a scan-based map that is aligned with the floor plan and usable for relative localization. This map improves the robustness of the system against unmapped obstacles and significant occlusion caused by large structures or clutter

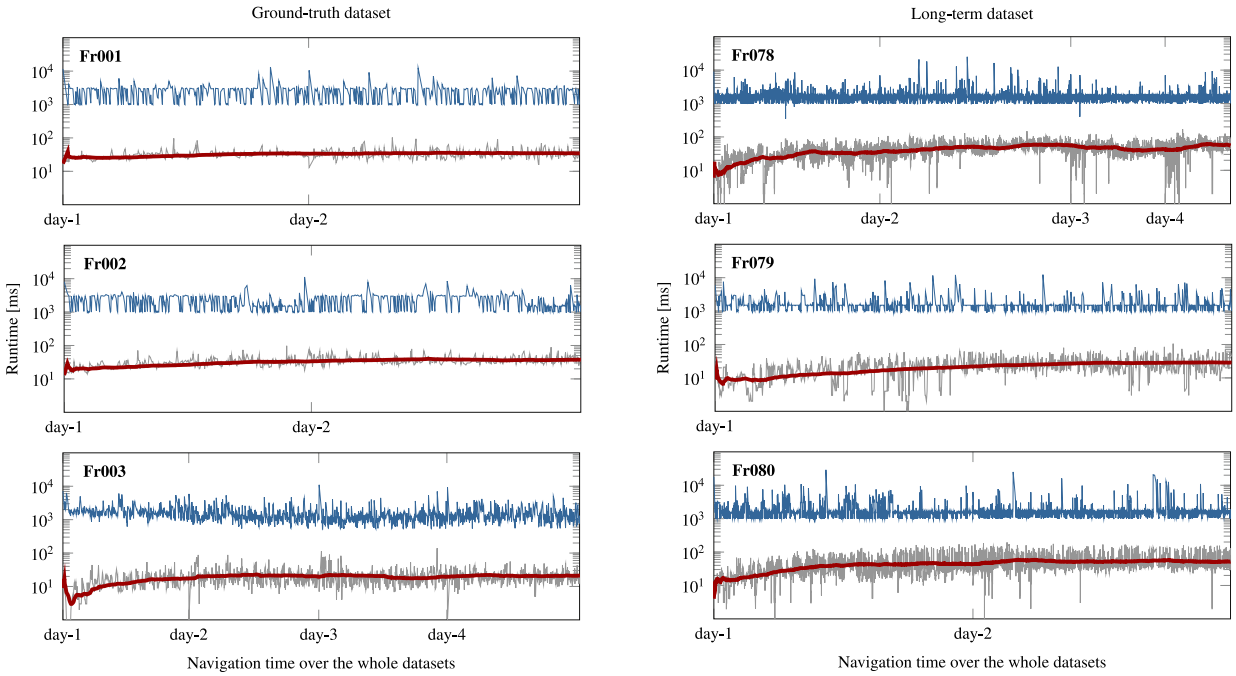


Fig. 6. Runtime of the localization method. In gray the system runtime, in red the moving average over a 5 min and 10 min window for the ground-truth and long-term datasets respectively. In blue the available time between two consecutive localization updates. The runtime remains bounded over time.

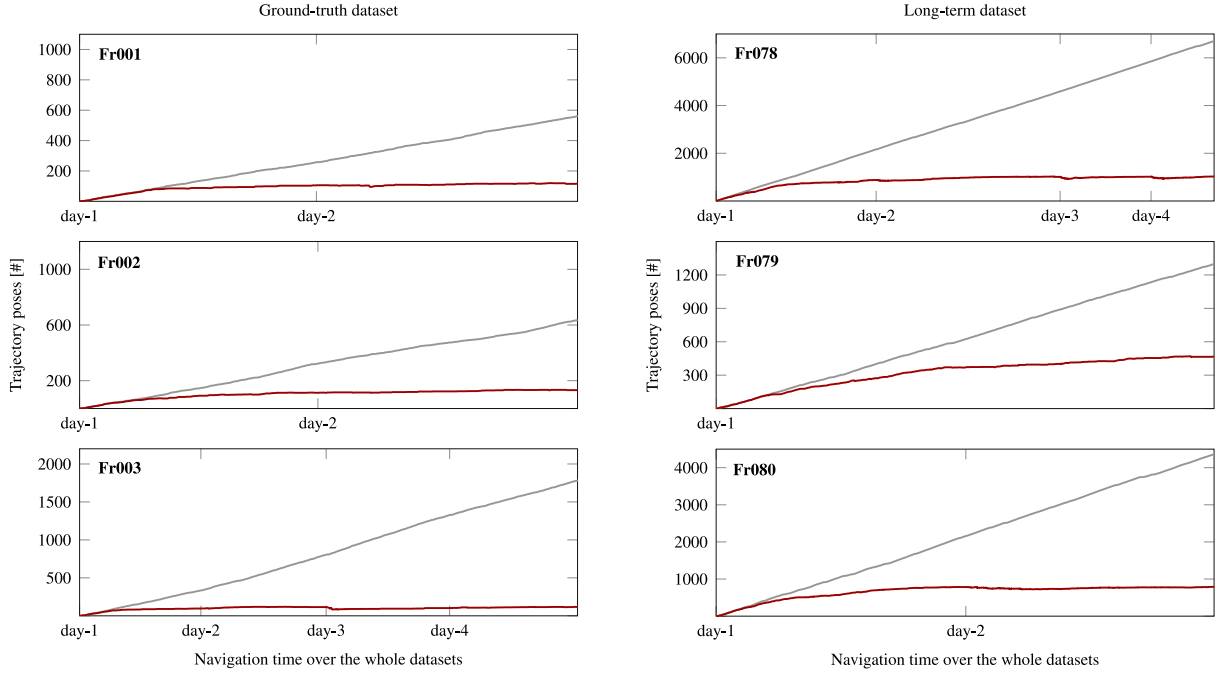


Fig. 7. Memory requirements of the localization method. The number of trajectory poses using the proposed mapping and pruning methods (in red) are compared to the number of trajectory poses obtained without pose graph sparsification (in gray). The size of the pose graph remains approximately stationary over time.

in the environment. In order to cope with long-term applications and to handle changing environments, we equipped our system with a robust front-end that estimates the probability for each node in the pose graph to store a LiDAR scan that is consistent with the scenario observable by the robot. Together with an efficient online pose graph pruning technique that bounds the memory requirements of the system, this allows the proposed method to run efficiently even over long periods.

The experimental evaluation shows that the proposed system works robustly in many real-world scenarios. Moreover, its localization accuracy is higher than state-of-the-art MCL algorithms even when these use sensor models that are robust to unmapped obstacles and clutter. In contrast to other approaches, the proposed method is capable of maintaining a consistent scan-map-based representation of the environment even when substantial reconfigurations prevent consistent loop closure measurements to be obtained using standard scan-matching techniques. Finally, the

memory consumption of the system remains bounded and real-time performance is achieved on average with commonly used sensors and platforms, which proves the usability of the system for long-term operation.

Acknowledgments

The authors are grateful to Christian Dornhege, University of Freiburg, and Slawomir Sander, KUKA, for the fruitful discussions.

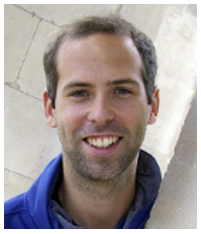
Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.robot.2018.11.003.

References

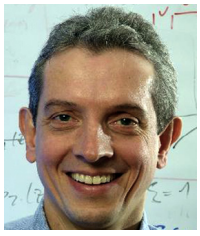
[1] A. Elfes, Sonar-based real-world mapping and navigation, IEEE J. Robot. Autom. 3 (3) (1987) 249–265.
[2] IEEE Standard for Robot Map Data Representation for Navigation, IEEE Standard 1873-2015, 2015, 1–54.
[3] L. Bowen-Biggs, S. Dazo, Y. Zhang, A. Hubers, M. Rueben, R. Sowell, W.D. Smart, C.M. Grimm, A method for establishing correspondences between hand-drawn and sensor-generated maps, in: Proc. of the International Conference on Social Robotics (ICSR), 2016.
[4] F. Boniardi, T. Caselitz, R. Kümmerle, W. Burgard, Robust LiDAR-based localization in architectural floor plans, in: Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017.
[5] R. Kümmerle, B. Steder, C. Dornhege, A. Kleiner, G. Grisetti, W. Burgard, Large scale graph-based SLAM using aerial images as prior information, Auton. Robots 30 (1) (2011) 25–39.
[6] R. Smith, M. Self, P. Cheeseman, Estimating uncertain spatial relationships in robotics, in: Autonomous Robot Vehicles, Springer, 1990, pp. 167–193.
[7] J.J. Leonard, H.F. Durrant-Whyte, Mobile robot localization by tracking geometric beacons, IEEE Trans. Robot. Autom. 7 (3) (1991) 376–382.
[8] W. Burgard, D. Fox, D. Hennig, T. Schmidt, Estimating the absolute position of a mobile robot using position probability grids, in: Proc .of the National Conference on Artificial Intelligence, 1996.
[9] D. Fox, S. Thrun, W. Burgard, F. Dellaert, Particle filters for mobile robot localization, in: Sequential Monte Carlo Methods in Practice, Springer, 2001, pp. 401–428.
[10] D. Fox, W. Burgard, S. Thrun, Markov localization for mobile robots in dynamic environments, J. Artificial Intelligence Res. 11 (1999) 391–427.
[11] S. Yilmaz, H.E. Kayir, B. Kaleci, O. Parlaktuna, Mobile robot localization via outlier rejection in sonar range sensor data, in: Proc. of the International Conference on Electrical and Electronics Engineering (ELECO), 2011.
[12] C. Sprunk, G.D. Tipaldi, A. Cherubini, W. Burgard, Lidar-based teach-and-repeat of mobile robot trajectories, in: Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013.
[13] M. Mazuran, F. Boniardi, W. Burgard, G.D. Tipaldi, Relative topometric localization in globally inconsistent maps, in: Robotics Research, Springer, 2018, pp. 435–451.
[14] A. Schiotka, B. Suger, W. Burgard, Robot localization with sparse scan-based maps, in: Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017.
[15] F. Lu, E. Milios, Globally consistent range scan alignment for environment mapping, Auton. Robots 4 (4) (1997) 333–349.
[16] S. Siddiqi, G.S. Sukhatme, A. Howard, Experiments in Monte-Carlo localization using wifi signal strength, in: Proc. of the International Conference on Advanced Robotics (ICAR), 2003.
[17] S. Ito, F. Endres, M. Kuderer, G.D. Tipaldi, C. Stachniss, W. Burgard, W-RGBD-D: floor-plan-based indoor global localization using a depth camera and WiFi, in: Proc. of the IEEE International Conference on Robotics and Automation (ICRA), 2014.
[18] W. Winterhalter, F. Fleckenstein, B. Steder, L. Spinello, W. Burgard, Accurate indoor localization for RGB-D smartphones and tablets given 2D floor plans, in: Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015.
[19] H. Hile, G. Borriello, Positioning and orientation in indoor environments using camera phones, IEEE Comput. Graphics Appl. (CG&A) 28 (4) (2008).
[20] R.C. Luo, Y.-C. Lin, C.-C.. Kao, Autonomous mobile robot navigation and localization based on floor plan map information and sensory fusion approach, in: Proc. of the IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), 2010.
[21] O. Mendez, S. Hadfield, N. Pugeault, R. Bowden, SeDAR-semantic detection and ranging: humans can localise without lidar, can robots? arXiv preprint arXiv:1709.01500.
[22] D. Meyer-Delius, J. Hess, G. Grisetti, W. Burgard, Temporary maps for robust localization in semi-static environments, in: Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2010.
[23] D. Meyer-Delius, M. Beinhofer, W. Burgard, Occupancy grid models for robot mapping in changing environments, in: Proc. of the AAAI Conference on Artificial Intelligence, 2012.
[24] G.D. Tipaldi, D. Meyer-Delius, W. Burgard, Lifelong localization in changing environments, Int. J. Robot. Res. 32 (14) (2013) 1662–1678.
[25] T. Krajník, J.P. Fentanes, J.M. Santos, T. Duckett, Fremen: Frequency map enhancement for long-term mobile robot autonomy in changing environments, Trans. Robot. 33 (4) (2017) 964–977.
[26] P. Biber, T. Duckett, et al., Dynamic maps for long-term operation of mobile service robots, in: Proc. of Robotics: Science and Systems, 2005.
[27] A. Walcott-Bryant, M. Kaess, H. Johannsson, J.J. Leonard, Dynamic pose graph SLAM: Long-term mapping in low dynamic environments, in: Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2012.
[28] D.M. Rosen, J. Mason, J.J. Leonard, Towards lifelong feature-based mapping in semi-static environments, in: Proc. of the IEEE International Conference on Robotics and Automation (ICRA), 2016.
[29] J. Van De Ven, F. Ramos, G.D. Tipaldi, An integrated probabilistic model for scan-matching, moving object detection and motion estimation, in: Proc. of the IEEE International Conference on Robotics and Automation (ICRA), 2010.
[30] D.Z. Wang, I. Posner, P. Newman, Model-free detection and tracking of dynamic objects with 2D lidar, Int. J. Robot. Res. 34 (7) (2015) 1039–1063.
[31] J. Röwekämper, C. Sprunk, G.D. Tipaldi, C. Stachniss, P. Pfaff, W. Burgard, On the position accuracy of mobile robot localization based on particle filters combined with scan matching, in: Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2012.
[32] A.N. Walcott, Long-term Robot Mapping in Dynamic Environments (Ph.D. thesis), Massachusetts Institute of Technology, Cambridge, MA, USA, 2011.
[33] G. Grisetti, R. Kümmerle, C. Stachniss, W. Burgard, A tutorial on graph-based SLAM, Intell. Transp. Syst. Mag. 2 (4) (2010) 31–43.
[34] A. Segal, D. Haehnel, S. Thrun, Generalized-ICP, in: Proc. of Robotics: Science and Systems, 2009.
[35] O. Vysotska, C. Stachniss, Exploiting building information from publicly available maps in graph-based SLAM, in: Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016.
[36] S. Thrun, W. Burgard, D. Fox, Probabilistic Robotics, MIT Press, 2005.
[37] J.C. Hart, Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces, Vis. Comput. 12 (10) (1996) 527–545.
[38] W. Donnelly, Per-pixel displacement mapping with distance functions, GPU gems 2 (22) (2005) 3.
[39] H. Kretzschmar, C. Stachniss, Information-Theoretic pose graph compression for Laser-based SLAM, Int. J. Robot. Res. 31 (2012) 1219–1230.
[40] N. Carlevaris-Bianco, R.M. Eustice, Generic factor-based node marginalization and edge sparsification for pose-graph slam, in: Proc. of the IEEE International Conference on Robotics and Automation (ICRA), 2013.
[41] G. Huang, M. Kaess, J.J. Leonard, Consistent sparsification for graph optimization, in: Proc. of the IEEE European Conference on Mobile Robots (ECMR), 2013.
[42] M. Mazuran, W. Burgard, G.D. Tipaldi, Nonlinear factor recovery for long-term SLAM, Int. J. Robot. Res. 35 (1–3) (2016) 50–72.
[43] R. Kümmerle, M. Ruhnke, B. Steder, C. Stachniss, W. Burgard, Autonomous robot navigation in highly populated pedestrian zones, J. Field Robot. 32 (4) (2015) 565–589.
[44] E.B. Olson, Robust and Efficient Robotic Mapping (Ph.D. thesis), Massachusetts Institute of Technology, Cambridge, MA, USA, 2008.
[45] F. Lu, E. Milios, Robot pose estimation in unknown environments by matching 2d range scans, J. Intell. Robot. Syst. 18 (3) (1997) 249–275.
[46] J. Hopcroft, R. Tarjan, Algorithm 447: efficient algorithms for graph manipulation, Commun. ACM 16 (6) (1973) 372–378.
[47] P. Agarwal, G.D. Tipaldi, L. Spinello, C. Stachniss, W. Burgard, Robust map optimization using dynamic covariance scaling, in: Proc. of the IEEE International Conference on Robotics and Automation (ICRA), 2013.
[48] N. Sünderhauf, P. Protzel, Switchable constraints for robust pose graph SLAM, in: Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2012.
[49] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, W. Burgard, g^2o: A general framework for graph optimization, in: Proc. of the IEEE International Conference on Robotics and Automation (ICRA), 2011.
[50] A. Censi, An accurate closed-form estimate of icp's covariance, in: Proc. of the IEEE International Conference on Robotics and Automation (ICRA), 2007.
[51] A. Censi, An ICP variant using a point-to-line metric, in: Proc. of the IEEE International Conference on Robotics and Automation (ICRA), 2008.
[52] H. Scharr, Optimale Operatoren in der digitalen Bildverarbeitung (Ph.D. thesis), Universität Heidelberg, 2000.

Federico Boniardi studied Mathematics at the University of Milan and Artificial Intelligence at the University of Edinburgh. Currently, he is a Ph.D. student at the Laboratory for Autonomous Intelligent Systems of the University of Freiburg headed by Wolfram Burgard. His research interests include robot localization, mapping and navigation.



Tim Caselitz is a Ph.D. student at the Laboratory for Autonomous Intelligent Systems of the University of Freiburg, which is headed by professor Wolfram Burgard. He received his diploma degree in Electrical Engineering and Information Technology from Karlsruhe Institute of Technology in 2013. His research is focused on robotic perception using camera, RGB-D, and LiDAR sensors.

Rainer Kümmerle is currently working as Product Owner at KUKA. He received his Ph.D. degree from the University of Freiburg in April 2013 where he was working in the Laboratory for Autonomous Intelligent Systems of the University of Freiburg headed by Wolfram Burgard. His research interests lie in the areas of navigation, mapping, and localization.



Wolfram Burgard is a professor for computer science at the University of Freiburg, Germany where he heads the Laboratory for Autonomous Intelligent Systems. He received his Ph.D. degree in computer science from the University of Bonn in 1991. His areas of interest lie in artificial intelligence and mobile robots. In the past, Wolfram Burgard and his group developed several innovative probabilistic techniques for robot navigation and control. They cover different aspects such as localization, map building, path-planning, and exploration. For his work, Wolfram Burgard received several best paper awards from outstanding national and international conferences. In 2009, Wolfram Burgard received the Gottfried Wilhelm Leibniz Prize, the most prestigious German research award. In 2010 he received the Advanced Grant of the European Research Council. Wolfram Burgard is the spokesperson of the Cluster of Excellence BrainLinksBrainTools and President of the IEEE Robotics and Automation Society.