2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Resource-Aware Collaborative Monte Carlo Localization with Distribution Compression

Nicky Zimmerman Alessandro Giusti Jérôme Guzzi

***Abstract*— Global localization is essential in enabling robot autonomy, and collaborative localization is key for multi-robot systems, allowing for more efficient planning and execution of tasks. In this paper, we address the task of collaborative global localization under computational and communication constraints. We propose a method which reduces the amount of information exchanged and the computational cost. We also analyze, implement and open-source seminal approaches, which we believe to be a valuable contribution to the community. We exploit techniques for distribution compression in near-linear time, with error guarantees. We evaluate our approach and the implemented baselines on multiple challenging scenarios, simulated and real-world. Our approach can run online on an onboard computer. We release an open-source C++/ROS2 implementation of our approach, as well as the baselines.**[1]

I. Introduction

Globally localizing in a given map is essential for enabling robot autonomy. Localization becomes extremely challenging when the environment is highly symmetric, featureless or very dynamic. Human-oriented indoor environments such as office buildings, contain high degree of geometric symmetry due to repetitive structures. In addition, the readily-available map representations such as floor plans are lacking in details, resulting in seemingly identical structures. Relying solely on geometric features may result in localization failure, leading researchers to exploit additional sources of information. RFID [19] and Wi-Fi signal strength [18] can be used to improve pose estimation, as well as textual cues [37], [7]. Another venue is utilizing semantic information [1], [16], [19], harnessing the significant progress in the fields of semantic understanding. However, single-robot localization can still fail, and recovering from a localization failure in the single robot scenario mostly involves a human in the loop, in particular when navigating with erroneous pose estimation raises safety concerns. In contrast, in a multi-robot setup, a poorly-localized robot can recover if assisted by a well-localized robot.

In the last decade, multi-robot systems have become more prevalent, introducing a new challenge of multi-robot localization. As opposed to single-robot localization, where a robot estimates its pose based on its own sensing, in a collaborative setting a robot can make use of information it receives from other agents. A recurring pattern in the related research is using the state estimation of one robot to improve the localization of another robot upon detection [2], [11], [26]. This information exchange involves broadcasting the belief of the robots, a distribution that is often approximated by a particle filter in the case of global localization. When the area the robots operate in is large, the particle set representing the belief can easily reach tens of thousands of particles. The naive approach of simply broadcasting all of the particles and then integrating them into another robot's belief [26], is computationally costly and has high bandwidth consumption. Therefore, a compressed representation of the belief is imperative and has been proposed in the past [26]. In this paper, we analyse the computational complexity related to compressing, exchanging, and fusing robots' beliefs for collaborative localization.

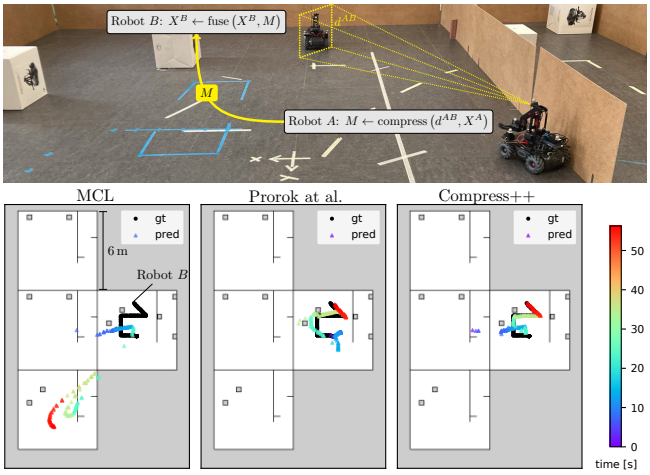


Fig. 1: Top: two robots during one experimental run when robot A detects robot B. Bottom: localization of robot B in the same run using 3 methods; prediction, color-coded for time, is plotted against ground truth position (black). Two failures, using non-collaborative Monte Carlo localizartion (MCL) [8] and Prorok et al. [26], and a successful convergence after 20 s with our collaborative localization approach (Compress++).

The main contribution of this paper is a novel approach to collaborative global localization (Fig. 1) which reduces the amount of data communicated and the computational cost. Additionally, we provide a unified overview and thorough analysis of alternative approaches to compress belief exchange. Furthermore, we release an open-source C++/ROS2 implementation for seminal works, as baselines for collaborative localization. While the methods are capable of operating in larger multi-robot scenarios, our experiments focus on the case of two collaborating robots, where we show that our ap-

All authors are with the Dalle Molle Institute for Artificial Intelligence (IDSIA), USI-SUPSI. This work was supported in part by REXASI-PRO H-EU project, call HORIZON-CL4-2021-HUMAN-01-01, Grant agreement 101070028. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the funding agencies, which cannot be held responsible for them.

[1] https://github.com/idsia-robotics/Collaborative-Monte-Carlo-Localization

proach (i) improves collaborative localization, (ii) decreases the required bandwidth, (iii) reduces the computational load, (iv) runs online on an onboard computer.

The paper is organized as follows. Sec. II provides a summary of related research in collaborative localization. In Sec. III, we present our novel approach for collaborative global localization with distribution compression, as well as overview of alternative approaches. We analyze the complexity of distribution compression, communication and fusion in Sec. IV. We introduce our experimental setup in Sec. V. In Sec. VI, we evaluate the performance of our approach against several baselines, and offer insights regarding their different behaviors.

II. Related Work

Map-based localization is an integral part of enabling the autonomy of mobile robots [5], [33]. Global localization for a single robot is widely-researched, and probabilistic methods [8], [12], [33] have gained popularity due to their robustness. The growing interest in multi-robot systems, presented a new challenge of collaborative multi-robot localization. While the term multi-robot localization sometimes refers to relative positioning [17], [20], we focus on cooperative global localization in a given map [11].

Multi-centralized approaches [22], [23], [30], where each robot maintains the belief for all robots, generally do not scale well with the number of robots. In decentralize approaches, each robot estimates only its own state and integrates relative observations from other robots when available.

Similarly to the single robot case [1], [16], [19], [35], [36], leveraging semantic scene understanding [6], [15], [32], [4] was adopted for multi-robot localization. A common approach to collaborative localization relies on robot detection, where one robot can sense another robot. Fox et al. [11] propose a factorial representation where each robot maintains its own belief, and the belief of different robots are assumed to be independent from each other. When one robot detects another, the detection model is used to synchronize their beliefs. However, no analysis is provided about the processing requirements and the experiments only cover one scenario. Barea et al. [2], propose a system for collaborative localization based on Monte-Carlo localization [8] framework, but do not include an explicit detection model.

Wu et al. [34] present an improvement to Fox et al. [11], where they consider whether the belief should be updated upon detection, by comparing the entropy of both robots' beliefs. They too, do not provide information about the computational and bandwidth requirement for the information exchange. Özkucur et al. [25] introduce an approach to collaborative localization where robots can be detected but not identified, but no details are given about the complexity of the approach. Prorok et al. [27] first detail a naive approach of synchronizing the beliefs of two robots, each with N particles, with $O(N^2)$ complexity. In their later work [26], they provide a clustering algorithm to reduce the complexity to $O(NK)$, where K is the user-defined number of clusters. In both works by Prorok et al. [26], [27], the experimental setup does not include exteroceptive sensing and the utilization of a map, while we explore the contribution of belief exchange in the framework of a range-based Monte Carlo localization (MCL). With the exception of Prorok's works, the constrains imposed by limited compute and bandwidth are largely unaddressed. Furthermore, the cited works present limited results for their individual approach, without comparing them against other baselines. In our work, we provide thorough analysis for the computational and communication requirements of seminal works [11], [26], as well as present resource-aware alternative detection models. We benchmark the various approaches on several environments, and offer insights regarding their performance.

III. Approach

We aim to globally localize a team of robots in a given map. In Sec. III-A, we give a brief overview of Monte Carlo localization [8]. We describe in Sec. III-B a common approach to distributed multi-robot localization, where beliefs are exchanged when robots detect each other, including the concept of reciprocal sampling. In Sec. III-C, we introduce our novel approach and, in Sec. III-D, we give an overview of alternative methods.

A. Monte Carlo Localization

Monte Carlo localization [8] is a recursive Bayesian filter that estimates a robot's state $\boldsymbol{x}_t$ at discrete time $t \in \mathbb{N}$, given map m, odometry readings $\boldsymbol{u}_t$, and sensor readings $\boldsymbol{z}_t$. We define the robot's state $\boldsymbol{x} = (x, y, \theta)^\top$ as an horizontal position $(x, y) \in \mathbb{R}^2$ and orientation $\theta \in [0, 2\pi)$; the map m is an occupancy grid map [21]; and the sensor measurement $\boldsymbol{z} \in \mathbb{R}^K_{\geq 0}$ has K beams. A particle filter represents the robot's belief $p(\boldsymbol{x}_t | \boldsymbol{u}_{1:t}, \boldsymbol{z}_{1:t}, m)$ as a set $X_t = \{s_{t,i}, i = 1, \ldots, N\}$ where each particle $s_{t,i} = (\boldsymbol{x}_{t,i}, w_{t,i})$ assigns weight $w_{t,i}$ to a sample state $\boldsymbol{x}_{t,i}$.

The proposal distribution $p(\boldsymbol{x}_t | \boldsymbol{x}_{t-1}, \boldsymbol{u}_t)$ is sampled when odometry $\boldsymbol{u}_t$ is available, using a motion model that considers the robot' kinematics and odometry noise σ_{odom}. For each observation, the particle filter is updated based on the sensor model, where weight is assigned to each particle according to the likelihood of the observation given its state, i.e., $w_{t,i} = p(\boldsymbol{z}_t | \boldsymbol{x}_{t,i}, m)$. As observation model $p(\boldsymbol{z} | \boldsymbol{x}, m)$, we use a beam-end model [33] with standard deviation σ_{obs}. We perform the update only when the robot moves at least for a trigger distance $(\delta_{xy}, \delta_\theta)$ to avoid repeated integration of the same observation. In the resampling step, particles are selected based on their assigned weight: we resample a particle set of size N using low-variance resampling [33] with an efficiency coefficient $N/2$.

B. Collaborative Monte Carlo Localization

Collaborative Monte Carlo localization extends the single robot MCL (Sec. III-A) by incorporating information from other robots into a robot's belief. Each robot locally runs an independent MCL that integrates its own sensor and odometry readings. As illustrated in Fig. 1, when robot A detects

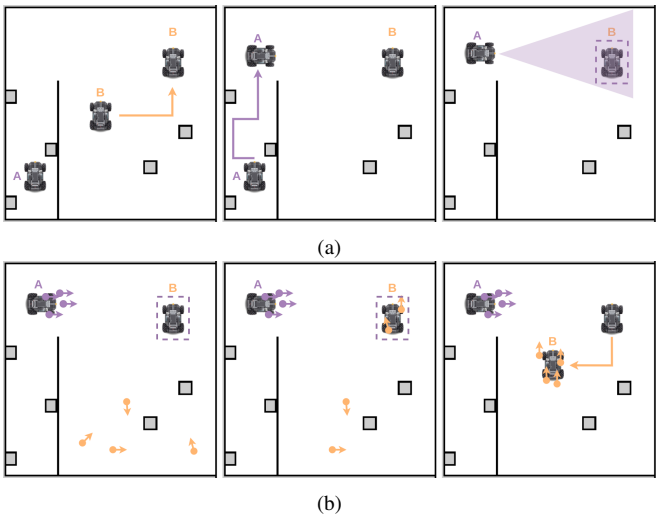


Fig. 2: (a) An illustration of an experimental run up to the first detection event. (b) A run where robot B is has no particles around its truth position at the time of detection (left), followed by reciprocal sampling (center) and successful localization (right).

robot B at time t, it estimates the *relative* position $d_t^{AB} \in \mathbb{R}^2$ of B and then sends message $M = M\left(d_t^{AB}, X_t^A\right)$ to B which carries (possibly compressed) information about its state X_t^A and detection d_t^{AB}; B, upon receiving the message, updates the weights of its particles [11], fusing information:

$$w_{t,i}^B = w_{t-1,i}^B p\left(\boldsymbol{x}_{t,i}^B | M\left(d_t^{AB}, X_t^A\right)\right). \tag{1}$$

A straightforward but naive way of implementing the right-most factor in Eq. (1) is to include the entire particle set X_t^A in the message $M = (d_t^{AB}, \boldsymbol{x}_t^A)$. Then

$$p(\boldsymbol{x}_{t,i}^B | d_t^{AB}, X_t^A) = \sum_{j=1}^{N} w_{t,j}^A p(\boldsymbol{x}_{t,i}^B | d_t^{AB}, \boldsymbol{x}_{t,j}^A), \tag{2}$$

where $p(\boldsymbol{x}^B | d^{AB}, \boldsymbol{x}^A)$ is the detection model, which we model as a normal distribution with variance Σ (detection noise) around a position corresponding to d^{AB}

$$p\left(\boldsymbol{x}^B | d^{AB}, \boldsymbol{x}^A\right) = \mathcal{N}\left((x,y)^B; T(d^{AB}; \boldsymbol{x}^A), \Sigma\right), \tag{3}$$

where $T(\cdot; \boldsymbol{x}^A)$ transforms a relative position with respect to pose $\boldsymbol{x}^A$ to an absolute position; please note that detections have no information about orientation. This naive implementation has quadratic time-complexity (see Section IV), which is problematic when resources are limited. For this reason, in Sec. III-C, we compress the belief of A in M, summarizing it with fewer representative points.

1) Reciprocal Sampling: To accelerate convergence, Prorok et al. [26], during the particle filter resampling step, propose sampling from the detection distribution (i.e., the right-most factor in Eq. (1)) with probability $\alpha > 0$. In essence, particles in the detected robot's filter are replaced with particles suggested by the detecting robot, based on its belief and the estimated relative position. The reciprocal sampling procedure is illustrated in Fig. 2b.

C. Distribution Compression

We present our approach for near-linear time distribution compression, based on Compress++ [31]. This compression method performs better than standard thinning algorithms, such as independent and identically distributed (i.i.d.) sampling, which are not concise and have a large integration error [10]. An important measure is the maximum mean discrepancy (MMD) [13], which measures distance between two distribution or sample sets, as a difference between mean embedding of features. The MMD between two sample sets X_1 and X_2 is defined as

$$\mathrm{MMD}(X_1, X_2) \doteq ||\mathbb{E}_{X_1}[K(X_1)] - \mathbb{E}_{X_2}[K(X_2)]||_{\mathcal{H}}, \tag{4}$$

where $k(\cdot)$ is the reproducing kernel, such that $K(X) \doteq (k(x_i, x_j))_{ij}$ is a symmetric positive semi-definite matrix over all input points $x_i \in X$. Kernel thinning (KT) algorithms [9], [10] use a better than i.i.d., non-uniform randomness to thin a sample set. KT algorithms recursively partition the input into balanced coresets, by ensuring each pair of coresets minimizes the MMD. In the initial step, there are two empty coresets, and two samples, x and x', from the original sample set are chosen at random. The assignment of each of the samples to a coreset is designed to minimize $\mathrm{MMD}(X_1 \cup \{x\}, X_2 \cup \{x'\})$. This step is repeated until all samples from the original set are assigned to one of the two coresets. This yields a near-optimal thinning procedure, which compresses a set of points while providing error guarantees. However, it suffers from quadratic or super-quadratic runtime. Shetty et al. [31] introduce Compress++, a meta-procedure for speeding up thinning algorithms while suffering at most a factor of 4 in error. This root-thinning algorithm returns a subset of $\sqrt{N}$ samples, with time complexity of $O(N \log^3 N)$.

In our formulation, we first compute a set

$$\bar{X}_t^{AB} = \{T(d_t^{AB}; \boldsymbol{x}^A) | (\boldsymbol{x}^A, \cdot) \in X_t^A\} \subset \mathbb{R}^2 \tag{5}$$

of *position* samples for B from the particles set X^A (after resampling, i.e., when particles have uniform weights) and then compress it using Compress++ to $\tilde{X}_t^{AB} \subset \bar{X}_t^{AB}$: this representative subset is then used in Eq. (2) and Eq. (3) instead of the whole set. We adopt the idea of reciprocal sampling for our distribution compression, by first sampling uniformly from $\tilde{X}_t^{AB}$ and then sampling a pose using Eq. (3):

$$\mu \sim \mathcal{U}(\tilde{X}_t^{AB}), \quad (x,y)_{t,i}^B \sim \mathcal{N}(\mu, \Sigma), \quad \theta_{t,i}^B \sim \mathcal{U}([0, 2\pi)).$$

D. Baselines

As part of our contribution, we implement several baselines, including seminal works [11], [26] in collaborative localization. We provide an overview of these approaches, using common notations for clarity.

1) Density Estimation Tree: Fox et al. [11] propose to use density estimation trees (DET) [24], [28] to transform the sample set into a piece-wise constant density function. First, DET^{AB} is constructed from $\bar{X}_t^{AB}$ and shared with B, who then updates its belief by querying it:

$$p\left(\boldsymbol{x}_{t,i}^B | M\left(d_t^{AB}, X_t^A\right)\right) = \mathrm{DET}^{AB}((x,y)_{t,i}^B). \tag{6}$$

While DET was suggested as a remedy for the nontrivial issues of establishing correspondence between two sample sets (X_t^B and $\bar{X}_t^{AB}$) without an explicit detection model, it can be used to compress the distribution by limiting the size T of the tree. For this method, we do not perform reciprocal sampling because the implementation detailed in the paper does not include it.

2) Divide-and-Conquer Clustering: Prorok et al. [26] propose a non-iterative, order-independent, non-parametric clustering inspired by multidimensional binary trees [3]. The particles of A are clustered into K cluster abstractions, which include the centroid $\boldsymbol{c}_{t,k}^A$, weight w_k^A, detection mean μ_k^A and detection variance Σ_k^A. The belief is updated as

$$p\left(\boldsymbol{x}_{t,i}^B \middle| M\left(d_t^{AB}, X_t^A\right)\right) = \sum_{k=1}^{K} w_k^A \mathcal{N}(T^{-1}\left((x,y)_{t,i}^B; \boldsymbol{c}_{t,k}^A\right); \mu_k^A, \Sigma_k^A + \Sigma),$$

where T^{-1} transforms absolute to relative positions, the multivariate normal distribution is represented in relative polar coordinates, and Σ captures the detection noise.

3) K-means Clustering: K-means clustering [14] is a commonly-used, low-cost iterative clustering method. It is mentioned by Prorok et al. [26] in the context of collaborative localization, where the authors dismiss it as too sensitive to the initial cluster assignment, but no comparison is reported. We introduce a belief compression based on K-means clustering: we compute K clusters of $\bar{X}_t^{AB}$ with centroids $c_{t,k}^A$; for each cluster, we compute its intra-cluster variance Σ_k^A and total weight w_k^A. The belief is then updated as

$$p\left(\boldsymbol{x}_{t,i}^B \middle| M\left(d_t^{AB}, X_t^A\right)\right) = \sum_{k=1}^{K} w_k^A \mathcal{N}\left((x,y)_{t,i}^B; c_{t,j}^A, \Sigma_k^A + \Sigma\right) \tag{7}$$

By clustering $\bar{X}_t^{AB}$ instead of X^A, we reduce the amount of information to broadcast compared to the approach suggested by Prorok et al. [26]. For the reciprocal sampling strategy, we sample from the mixture of normal distributions of Eq. (7).

4) Standard Thinning: The most common approach to reduce the number of samples is i.i.d. sampling, where K samples are randomly picked from a set of size N. This can be used to reduce $\bar{X}^{AB}$, share it and then apply Eq. (2). In our implementation, we also include a reciprocal sampling step similar to the one preformed for our Compress++ approach.

IV. Complexity Analysis

The complexity of all approaches is presented in Tab. I. We discuss the complexity of 3 components: compression, communication, and fusion. These time and space complexities become significant in systems with many robots or when the computational and communication resources are constrained.

A. Compression

The naive implementation for belief exchange requires just $O(N)$ complexity for compression, but results in high communication cost and computational cost during fusion.

TABLE I: Algorithm Complexity. N is the number of particles. K is the number of clusters for Prorok et al. and K-means, and the number of points selected by standard thinning.

Method	Naive	Std. Thinning	Fox et al.	Prorok et al.	K-means	Compress++
Compression	$O(N)$	$O(K)$	$O(HDN \log N)$	$O(NK)$	$O(NKL)$	$O(N \log^3 N)$
Communication	$O(N)$	$O(K)$	$O(N)$	$O(K)$	$O(K)$	$\sqrt{N}$
Fusion	$O(N^2)$	$O(NK)$	$O(N \log N)$	$O(NK)$	$O(NK)$	$O(N\sqrt{N})$

The construction of a DET involves leave-one-out cross-validation, resulting in a $O(HDN \log N)$ complexity for N data samples with D features and H tries. For the non-iterative clustering method suggested by Prorok et al. [26], the complexity for K clusters is $O(NK)$. The complexity of K-means clustering is $O(NKL)$, where L in the number of iterations. Even though K-means clustering is an iterative approach, for $L = 5$ this compression strategy is comparable to Prorok et al.'s approach, and in practice runs faster (see Sec. VI-C). The complexity of the standard thinning algorithm is $O(K)$, where K is the size of the reduced sample set. To compress a state with N particles, our Compress++ approach has a time complexity of $O(N \log^3 N)$.

B. Communication

When multiple robots communicate on the same network, the total bandwidth required needs to be considered. The naive approach communicates the robot's belief by broadcasting all particle in the filter, which takes $O(N)$ space. To reduce the amount of information we broadcast, it is necessary to compress the belief. When the belief is represented as a DET ([11]), the information sent is in the order of $O(N)$, with constant coefficient that represents the space required for the bookkeeping of a single tree node. When the size of the tree is limited to T nodes, the cost is reduced to $O(T)$. Both Prorok et al.'s and K-means clustering use cluster representatives to summarize the belief, resulting in $O(K)$ space for K clusters. Similarly, for standard thinning, we only broadcast the reduced set of K particles. Using our Compress++ approach, the coreset representation reduces the communication cost to $O(\sqrt{N})$ ($K = \sqrt{N}$ in this case).

C. Fusion

Following Eq. (2), the complexity of updating the belief is $O(N^2)$ for the naive implementation. Fox et al. [11] requires $O(N \log N)$, since it involves querying a tree. The update step, combined with the reciprocal sampling, detailed by Prorok et al. [27] requires $O(NK)$ time; the same for K-means and std. thinning. Our Compress++ approach has also a $O(N\sqrt{N})$ complexity when updating the belief ($K = \sqrt{N}$ in this case).

V. Experimental Setup

We restrict the study to the case of two robots, although all presented methods generalize to larger multi-robot systems. In our experiments, robot B is delocalized at the time when it is first detected by robot A; we explore the contribution of information exchange to the localization performance of robot B.

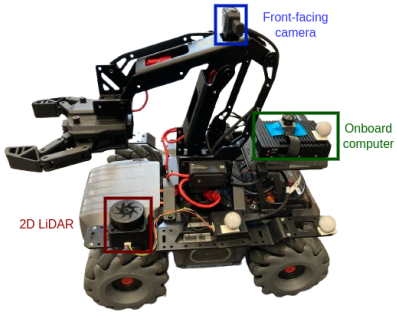


Fig. 3: The robotic platform used in the evaluation.

A. Robots

The platform we use in the evaluation is the DJI RoboMaster EP, a commercially available ground robot, with omnidirectional drive, and a size of $32\,\mathrm{cm} \times 24\,\mathrm{cm} \times 27\,\mathrm{cm}$. Depicted in Fig. 3, the robot has a front-facing CMOS HD camera with a $102°$ horizontal FoV and a YDLIDAR Tmini Pro 2D LiDAR with a range of $12\,\mathrm{m}$, $360°$ FoV, and resolution of $0.54°$ at $6\,\mathrm{Hz}$. The onboard robot firmware includes an ML model for detecting other RoboMasters in the camera stream. The detector runs at about $5\,\mathrm{Hz}$ and returns a list of bounding boxes in image space, from which, using calibrated homography, we reconstruct the relative horizontal position of detected robots $d^{AB} = (r, \theta)$, with a zero-mean gaussian error ($\sigma_r \approx 0.05r, \sigma_\theta \approx 0.03\,\mathrm{rad}$) that depends on the range r. The robots carry a Single Computer Board (Khadas VIM4) that runs ROS2 drivers for the robot platform[2] and LiDAR.

The same platform is available in simulation[3] (CoppeliaSim [29]), where we replicate the same sensing error model and run the same ROS2 driver.

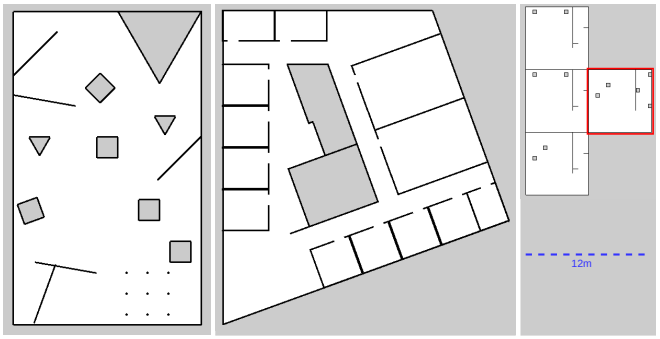
B. Environments

In simulation, we design three different environments (Fig. 4), with a varying degree of geometric symmetry and feature richness. Environment 1 is specifically designed to be feature-sparse, with multiple areas that challenge the limited range ($12\,\mathrm{m}$) of our LiDAR. Environment 2 is modeled after a floor of our building. Environment 3 is designed to have a large degree of symmetry. Environments 1 and 2 have a free area of $500\,\mathrm{m}^2$ where experiments are executed with 10000 particles. Environment 3 is smaller, with a free area of $140\,\mathrm{m}^2$ where experiments are executed with 2000 particles. To conduct real-world evaluation, we reconstruct the layout of the left room in environment 3 in our lab, which features a motion tracker system to collect ground truth information.

C. Scenario

We focus our experimental scenario on the impact of collaborative localization, therefore we make sure that each experimental run includes *multiple* detection events. We randomly choose start and goal poses of robot A and compute a shortest-path trajectory to follow. We then choose the start pose for robot B such that it would be seen by robot A at some point along its trajectory, and a random goal pose. In this scenario, robot B is delocalized at the time of the first detection by robot A, as illustrated in Fig. 2a.

[2]https://github.com/jeguzzi/robomaster_ros
[3]https://github.com/jeguzzi/robomaster_sim



(a) Env1 (b) Env2 (c) Env3

Fig. 4: Environments have varying degree of geometric symmetry and feature richness. The area highlighted in red was reconstructed in our lab for real-world evaluation. The three maps are to scale; the LiDAR range, 12 m, is marked in blue.

D. Metrics

Three metrics were considered for the evaluation - the success rate, absolute trajectory error (ATE) after convergence and convergence time. We define convergence as the time when the estimate pose is within $0.3\,\mathrm{m}$ radius (i.e., about twice the robot's size) of the ground truth pose, and the orientation is within $0.3\,\mathrm{rad}$. After convergence, the tracked pose must not diverge for an accumulated 5% of the remaining sequence. A localization run is successful if convergence is achieved in the first 90% of the sequence time and the tracked pose does not diverge. Each sequence is evaluated multiple times to account for the inherent stochasticity of the MCL framework.

E. Procedure and parameters

We first record odometry, LiDAR scans, detections and ground truth poses for each robots during all experimental runs. In total, we recorded 19 runs for the 3 environments in simulation and 17 runs for Env3 in real-world. Then, we evaluate each method presented in Sec. III on the same runs, using the parameters reported in Tab. II for the common MCL part. Overall, 112 evaluations were run for each of the 7 methods.

TABLE II: Common MCL parameters

σ_{odom}	σ_{obs}	$r_{\max}$	α	δ_{xy}	δ_θ
$(0.05, 0.05, 0.05)$	$0.5\,\mathrm{m}$	$12.0\,\mathrm{m}$	0.06	$0.05\,\mathrm{m}$	$0.05\,\mathrm{rad}$

The Compress++ algorithm reduces the sample set first to the closest power of 4, and then perform root-thinning, which resulted in 64 representative samples from a set of 10000 particles, and 32 representative samples for a sample set of 2000 particles. For Fox at al.'s approach, we ensured that we have no more than 20 leaves in the DET. Prorok et al. [27] explored different cluster numbers, between 1 and

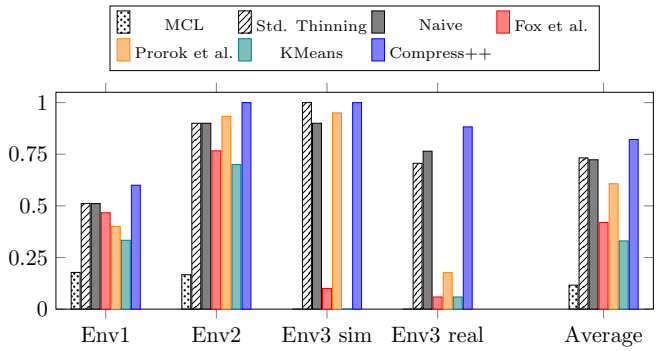


Fig. 5: The success rate of all methods for each of the environments for robot B.

TABLE III: Baseline comparison of global localization performance for robot B. We report success rate, convergence time in seconds (top) and ATE in $[\mathrm{rad}/m]$ format (bottom). ATE is not reported when all runs resulted in failure.

Method	Env1	Env2	Env3 (sim)	Env3 (real)	Average
MCL	17.8% (11.2)	16.7% (7.8)	0.0% (-)	0.0% (-)	11.6% (6.6)
Std. Thinning	51.1% (32.4)	90.0% (24.5)	**100.0%** (45.6)	70.6% (43.8)	73.2% (34.4)
Naive	51.1% (29.7)	90.0% (24.6)	90.0% (53.1)	76.5% (38.2)	72.3% (33.8)
Fox et al.	46.7% (34.8)	76.7% (17.4)	10.0% (91.1)	5.9% (39.5)	42.0% (40.9)
Prorok et al.	40.0% (43.3)	93.3% (18.3)	95.0% (41.0)	17.6% (27.5)	60.7% (33.8)
K-means	33.3% (51.9)	70.0% (16.5)	0.0% (-)	5.9% (39.9)	33.0% (31.3)
Compress++	**60.0%** (34.2)	**100.0%** (23.1)	**100.0%** (47.9)	**88.2%** (41.2)	**82.1%** (34.7)
MCL	0.032/0.198	0.005/0.175	-/-	-/-	0.014/0.127
Std. Thinning	0.026/0.199	0.022/0.103	0.031/0.062	0.062/0.153	0.031/0.142
Naive	0.026/0.201	0.021/0.106	0.023/0.071	0.060/0.186	0.029/0.150
Fox et al.	0.029/0.217	0.045/0.142	0.048/0.185	0.181/0.117	0.060/0.176
Prorok et al.	0.034/0.199	0.036/0.121	0.068/0.099	0.090/0.114	0.049/0.147
K-means	0.030/0.193	0.033/0.163	-/-	0.156/0.171	0.045/0.147
Compress++	0.024/0.214	0.018/0.108	0.026/0.062	0.060/0.154	0.028/0.149

32, and reported no significant change in performance for any $K > 1$. Therefore, for both Prorok et al.'s and the K-means approaches, we set $K = 8$. The reciprocal sampling ratio was set to $\alpha = 0.06$, as suggested by Prorok et al. [27].

VI. Experimental Evaluation

We conducted a thorough evaluation of the different approaches to cooperative localization. We present experimental results that support the claims that our proposed approach (i) improves collaborative localization, (ii) decreases the required bandwidth, (iii) reduces the computational load, (iv) runs online on an onboard computer.

A. Collaborative localization

We compare our approach against the different baselines. We focus on the impact of belief exchange between the somewhat-localized robot A and the delocalized robot B, on the localization performance of robot B, for the different approaches.

As can be seen from Fig. 5 and Tab. III, the results highlight the importance of reciprocal sampling. The seminal work by Fox et al. [11] does not include a reciprocal sampling step and performs poorly in many sequences. To support this claim, we evaluated the performance of the naive implementation with no reciprocal sampling, which result in a dramatic drop of performance, with success rates of (20%, 10%, 40%) in the 3 simulated environments respectively. As illustrate in Fig. 2b, reciprocal sampling is particularly crucial when robot B is delocalized and few-to-none particles are present around its true location, as reweighting particles in Eq. (1) has minimal impact: it can only encourage particles that are in the vicinity of the detection position, but not propose new hypotheses to robot B. In contrast, reciprocal sampling allows robot A to enrich robot B's particle filter with new samples.

Another interesting insight comes from the difference in performance in different environments. Due to the sparsity of Env1, robot A's localization is less accurate prior to the detection event. For the real-world environment, localization is challenging due to discrepancies between the map and the constructed maze. Additionally, the LiDAR is partially occluded by the arm, reducing the FoV to 300°. For these reasons, the localization for robot B is challenging even after integrating A's belief, as indicated by the higher ATE across all methods (Tab. III) in these two environments. Methods that perform well on the second and third (simulated) environment, like standard thinning and the approach of Prorok et al., suffer a significant loss in performance in the first environment, as well as in the real-world experiments. Our approach, Compress++, remains a top-performer in all 3 environments, including in the real-world, supporting our first claim. We provide a demonstration of our approach in a real environment in the attached video.

In Fig. 6, we visualize convergence as a function of time, where all sequences are synchronized such that $t = 0$ when the first detection message is received by robot B. For every time step, we report for what fraction of the runs the current pose estimation is below the convergence threshold. While in Fig. 5 and Tab. III success requires that pose estimation remains within a convergence threshold from the moment of convergence, in Fig. 6, we only consider whether a pose estimation is close enough to the ground truth at a given time. Therefore, Fig. 6 differs from results about success, yet shows a similar ranking between the methods, with Compress++ converging with the highest reliability. We note that while Prorok et al.'s approach converges well in the long run, it suffers particularly from instability as it tends to converge fast and then diverge, as seen in the drop around $t = 5\,\mathrm{s}$ and $t = 60\,\mathrm{s}$ in Env3 (real). Similar differences between convergence and success rate can be seen for all methods but to a lesser degree.

B. Bandwidth requirements

We go through the implementation of all methods to compute how much bandwidth they require. For the naive approach, where the entire distribution is exchanged, the size of a message is $12N = 120$ kB as each particle is represented by 3 floating point numbers, assuming 4-byte representation for floats. For the K-means approach, we require 6 floating point numbers to encode each cluster, i.e., a message size of $24K = 192$ bytes for $K = 8$ clusters. For Prorok et al.'s approach, we need 8 floating point numbers to describe each cluster abstraction, giving a message of 256 bytes. For Fox et al.'s, the whole DET is sent; since each node requires some bookkeeping (50 bytes per node, a conservative estimation), it requires $50T = 1$ kB for the entire tree when $T = 20$.

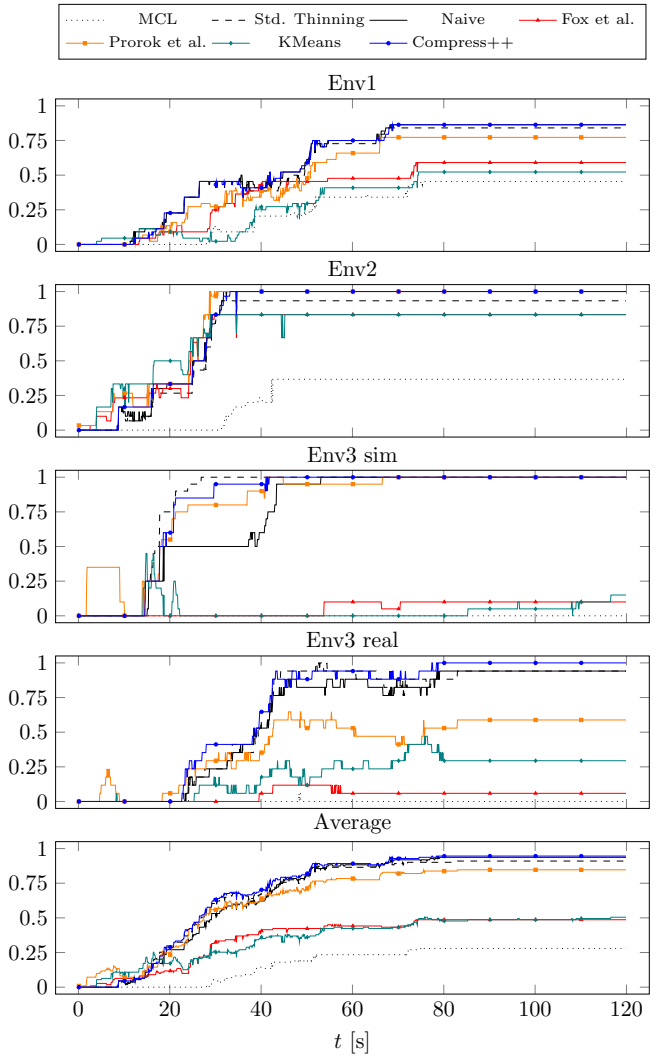


Fig. 6: The fraction of runs whose current pose estimation of robot B is below convergence threshold. The time $t = 0$ is synchronized by the arrival of the first detection message from robot A.

For standard thinning, we require $3K = 192$ bytes for $K = 64$ sampled particles. For Compress++, representative points have equal weight and are represented by 2 floats; as it first reduces the set to the nearest power of 4, and then takes the root, it requires 512 bytes for 10000 particles and 256 bytes for 2000 particles. Therefore our approach drastically reduces the bandwidth requirement compared to the naive approach (second claim), achieving a comparable compression rate as the other methods but maintaining a larger localization performance.

C. Runtime cost

As discussed in Sec. IV, our approach reduces the overall time-complexity compared to the naive approach: on one side, it increases the time-complexity of compression for robot A by factor $\log^3 N$, on the other side, it decreases the larger time-complexity for robot B by a more significant factor $\sqrt{N}$, supporting our third claim.

The runtime cost for all collaborative localization approaches is presented in Tab. IV. We benchmarked the approaches using 10000 particles. Since the output of Compress++ is 64 representative points, we select a comparable $K = 64$ for Prorok et al., std. thinning, and K-means, and constrain the DET construction to result in about 64 nodes. As expected, the naive approach requires the longest time to perform the belief fusion.

TABLE IV: Runtime cost in milliseconds for one update step of filters with 10000 particles.

HW	Method	Std. Thinning	Naive	Fox et al.	Prorok et al.	K-means	Compress++
Laptop	Compression (A)	0.1	0.1	5.4	20.8	1.6	32.2
	Fusion (B)	2.6	286.1	0.4	1.0	1.5	2.5
Khadas	Compression (A)	0.2	0.3	10.9	51	8.8	85
	Fusion (B)	9.9	1191.7	0.6	4.0	5.9	9.9

We evaluated the performance on two platform, a laptop with Intel Core i9 processors (16 cores), and Khadas VIM4 board with ARM Cortex processors (8 cores). Even though K-means clustering has a slightly higher theoretical complexity, for $L = 5$ this compression strategy is comparable to approach of Prorok et al., and in practice runs 12 times faster on the laptop and 5.8 times faster on the embedded platform. We attribute this performance gap to the fact that K-means is much easier to parallelize. While all other methods are implemented in C++ and are optimized for multi-core execution, our method utilizes an open-source Python implementation of Compress++.[4] Nonetheless, its runtime for compression is on-par with the approach of Prorok et al. and takes less than 10 ms to integrate the belief, supporting our fourth claim that it can run in real-time onboard. Additionally, the attached video shows how our localization method runs online, onboard of the robots.

D. Clustering

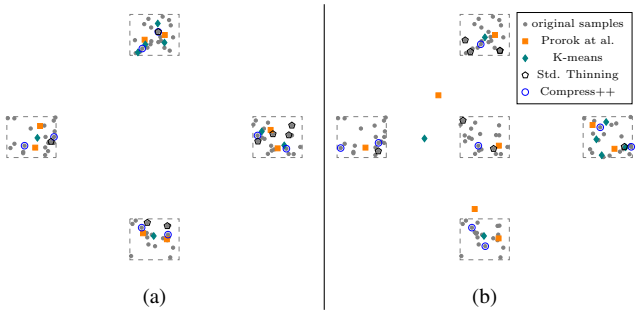


Fig. 7: The behavior of different distribution compression methods on different data points formations.

We explore an artificial yet interesting case of compression in the presence of symmetry, as would occur when a robot is not yet localized but maintains several hypothesis, e.g., due to geometric symmetry of the environment. We generated 20 points each from 4 evenly spaced regions in a diamond formation (Fig. 7a) and another 20 points from a region at the center (Fig. 7b). We then apply different methods to extracts 8 representative points.

[4] https://github.com/microsoft/goodpoints/tree/main

This test case reveals a weakness of the divide-and-conquer clustering algorithm proposed by Prorok et al. [26]: as shown in Fig. 7b, the algorithm struggles to divide the particle set into well-defined clusters, likely due to the division strategy, which separates clusters along the axis of highest variance. The algorithm also fails when an even number of clusters are places along a circle with an additional one in the middle.

K-means clustering also fails to perfectly segment the clusters on this type of points distribution. The standard thinning approach, i.e., sampling 8 points at random, is naturally less expressive than other density estimation approaches, yet, in this case, it represents 4 out of the 5 clusters. Compress++ succeeds to extract samples from each cluster.

VII. Conclusion

In this paper, we presented a novel approach to resource-aware collaborative global localization. We also provided a detailed overview of different distribution compression approaches, as well as a C++/ROS2 implementation. Additionally, we conducted a thorough complexity analysis and bench-marking for the various methods, which we also open-source for the benefit of the community. In the future, we would like to extend the experiments to larger groups of robots, as well as to expand the real-world experimental setup beyond a single room.

References

[1] N. Atanasov, M. Zhu, K. Daniilidis, and G.J. Pappas. Localization from semantic observations via the matrix permanent. *Intl. Journal of Robotics Research (IJRR)*, 35(1-3):73–99, 2016.

[2] R. Barea, E. López, L.M. Bergasa, S. Álvarez, and M. Ocaña. Collaborative multi-robot Monte Carlo localization in assistant robots. *Intl. Trans. on Systems Science and Applications*, 3(3):227–237, 2007.

[3] J.L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[4] A. Bochkovskiy, C.Y. Wang, and H.Y.M. Liao. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv preprint*, 2004.10934, 2020.

[5] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. Leonard. Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age. *IEEE Trans. on Robotics (TRO)*, 32:1309–1332, 2016.

[6] B. Cheng, M.D. Collins, Y. Zhu, T. Liu, T.S. Huang, H. Adam, and L.C. Chen. Panoptic-DeepLab: A Simple, Strong, and Fast Baseline for Bottom-Up Panoptic Segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[7] L. Cui, C. Rong, J. Huang, A. Rosendo, and L. Kneip. Monte-Carlo Localization in Underground Parking Lots Using Parking Slot Numbers. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2021.

[8] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 1999.

[9] R. Dwivedi and L. Mackey. Kernel Thinning. *arXiv preprint arXiv:2105.05842*, 2021.

[10] R. Dwivedi and L. Mackey. Generalized Kernel Thinning. In *Intl. Conf. on Learning Representations*, 2022.

[11] D. Fox, W. Burgard, H. Kruppa, and S. Thrun. A Probabilistic Approach to Collaborative Multi-robot Localization. *Autonomous Robots*, 8:325–344, 2000.

[12] D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research (JAIR)*, 11:391–427, 1999.

[13] A. Gretton, K.M. Borgwardt, M.J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

[14] J.A. Hartigan and M.A. Wong. A k-means clustering algorithm. *JSTOR: Applied Statistics*, 28(1):100–108, 1979.

[15] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2017.

[16] R. Hendrikx, P. Pauwels, E. Torta, H. Bruyninckx, and M. van de Molengraft. Connecting Semantic Building Information Models and Robotics: An application to 2D LiDAR-based localization. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021.

[17] A. Howard, M.J. Matark, and G.S. Sukhatme. Localization for mobile robot teams using maximum likelihood estimation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2002.

[18] S. Ito, F. Endres, M. Kuderer, G. Tipaldi, C. Stachniss, and W. Burgard. W-RGB-D: Floor-Plan-Based Indoor Global Localization Using a Depth Camera and WiFi. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2014.

[19] D. Joho, C. Plagemann, and W. Burgard. Modeling RFID signal strength and tag detection for localization and mapping. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2009.

[20] A. Martinelli, F. Pont, and R. Siegwart. Multi-robot Localization using Relative Observations. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2005.

[21] H.P. Moravec. Sensor Fusion in Certainty Grids for Mobile Robots. In *Sensor Devices and Systems for Robotics (SDSR)*, 1989.

[22] E.D. Nerurkar and S.I. Roumeliotis. Asynchronous multi-centralized cooperative localization. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[23] E.D. Nerurkar, S.I. Roumeliotis, and A. Martinelli. Distributed maximum a posteriori estimation for multi-robot cooperative localization. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2009.

[24] S. Omohundro. Bumptrees for efficient function, constraint and classification learning. In *Proc. of the Advances in Neural Information Processing Systems (NIPS)*, 1990.

[25] N.E. Özkucur, B. Kurt, and H.L. Akın. A collaborative multi-robot localization method without robot identification. In *RoboCup 2008: Robot Soccer World Cup XII 12*, 2009.

[26] A. Prorok, A. Bahr, and A. Martinoli. Low-cost collaborative localization for large-scale multi-robot systems. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2012.

[27] A. Prorok and A. Martinoli. A reciprocal sampling algorithm for lightweight distributed multi-robot localization. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[28] P. Ram and A.G. Gray. Density estimation trees. In *Proc. of the ACM/SIGKDD Intl. Conf. on Knowledge discovery and data mining*, 2011.

[29] E. Rohmer, S.P.N. Singh, and M. Freese. V-rep: A versatile and scalable robot simulation framework. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[30] S.I. Roumeliotis and G.A. Bekey. Distributed multi-robot localization. *IEEE Trans. on Robotics (TRO)*, pages 179–188, 2000.

[31] A. Shetty, R. Dwivedi, and L. Mackey. Distribution Compression in Near-linear Time. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2022.

[32] M. Sodano, F. Magistri, T. Guadagnino, J. Behley, and C. Stachniss. Robust double-encoder network for rgb-d panoptic segmentation. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2023.

[33] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[34] D. Wu and H. Su. An improved probabilistic approach for collaborative multi-robot localization. In *Proc. of the IEEE Intl. Conf. on Robotics and Biomimetics*, 2009.

[35] N. Zimmerman, T. Guadagnino, X. Chen, J. Behley, and C. Stachniss. Long-Term Localization Using Semantic Cues in Floor Plan Maps. *IEEE Robotics and Automation Letters (RA-L)*, 8(1):176–183, 2023.

[36] N. Zimmerman, M. Sodano, E. Marks, J. Behley, and C. Stachniss. Constructing Metric-Semantic Maps Using Floor Plan Priors for Long-Term Indoor Localization. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2023.

[37] N. Zimmerman, L. Wiesmann, T. Guadagnino, T. Läbe, J. Behley, and C. Stachniss. Robust Onboard Localization in Changing Environments Exploiting Text Spotting. *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2022.