Long-Term Indoor Localization in Floor Plans using Semantic Cues

for robot autonomy in human-oriented environments

Doctoral Dissertation submitted to the
Faculty of Informatics of the *Università della Svizzera italiana*
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

presented by

Nicky Zimmerman

under the supervision of

Prof. Kai Hormann

co-supervised by

Prof. Alessandro Giusti, Dr. Jérôme Guzzi, Prof. Luca Benini

September 2024

Dissertation Committee

Prof. Piotr Didyk	Università della Svizzera italiana, Switzerland
Prof. Luca Gambardella	Università della Svizzera italiana, Switzerland
Prof. Acquaviva Andrea	Università di Bologna, Italy
Prof. Sorrenti Domenico	Università di Milano Bicocca, Italy

Dissertation accepted on 24 September 2024

Prof. Kai Hormann
Research Advisor
Università della Svizzera italiana, Switzerland

Prof. Alessandro Giusti
Research Co-Advisor
Università della Svizzera Italiana, Switzerland

Dr. Jérôme Guzzi
Research Co-Advisor
Scuola universitaria professionale della Svizzera italiana, Switzerland

---

Prof. Luca Benini
Research Co-Advisor
ETH/Università di Bologna, Switzerland/Italy

Prof. Walter Binder & Prof. Stefan Wolf
PhD Program Director

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Nicky Zimmerman
Lugano, 24 September 2024

To Green Day, for teaching me how to march to my own beat.

"Hidden talent counts for nothing"

Nero

Acknowledgements

I would like to thank my main advisors, Alessandro and Jerome. I stormed back into their peaceful existence, and was met with a warm embrace. This is particularly special because they were well aware of all my flaws and I was not accounted for as part of their planned responsibilities.

I would like to express my sincere gratitude and appreciation to Prof. Cyrill Stachniss, for his guidance through mentoring and personal example of excellence in robotics research.

To my oldest friend, Dana Yudelevich, thank you for your saint's patience, listening to countless hours of me ranting. To Alex Gendelman, a manager-turned-friend, I'm grateful for your almost-optimistic perspective and solid confidence in my abilities.

A well-deserved mention goes to Hanna Müller, my collaborator since my M.Sc, when I started my robotics journey. Her wizard skills with hardware majorly contributed to the research presented in this thesis. I would also like to thank Prof. Luca Benini for supervising our collaboration, as well as Dr. Michele Magno and all of the excellent Beninis from IIS/PBL who helped along the way (Vlad, Georg, Tommaso, Julian,...).

Matteo Sodano, even though we had some fierce disagreements, I think we did well in overcoming our differences. Thank you for keeping me sane when I was at my lowest - to quote Prof. Stachniss "This environment is not enjoyable". I am also very proud of our collaborative research work. I would also like to acknowledge Prof. Marija Popovic, for her support and guidance even though I was not her responsibility, for her friendship even though we had a shaky start, and for being an inspiration for young researchers.

To my dear parents, I am sorry for the last 34 years. I would be happy to have you with me for at least another 34 years, and I will try to give you reasons to be proud of me for as long as you stick around.

Last but not least, my master directors, Prof. Kai Hormann and Prof. Evanthia Papadopoulou. Thank you for being inspirational figures, sympathetic ear and working relentlessly to improve the experience of students in the faculty. I am forever indebted to Prof. Kai Hormann for saving me from the hole I dag for myself, and providing me with a supportive academic environment to pursue my research.

Ah. And I would like to remind myself that I am capable and worthy, and a lot stronger than I usually think, to have made it to this point.

Abstract

Localization in a given map is an essential capability of most autonomous robots, and robust long-term localization is crucial in the case of service robots. This is a challenging task, especially in a dynamic, human-occupied environment, and it motivates the use of sparse map representations containing structural elements that remain constant over time. Floor plans, in particular, are a sparse map representation that is often readily-available without the additional cost and effort of sensor-based mapping.

We present different strategies for achieving robust long-term localization in floor plans, by taking inspiration from the way humans navigate in indoor environments. We start with a classical range sensor-based particle filter framework and augment it by integrating textual information. We then improve localization by considering a variety of semantic cues and propose a 3D metric-semantic map representation that enriches floor plans with semantic information. We address the challenge of localization on resource-constrained platforms and verify that our semantic localization approach is suitable for a variety of robotic platforms. Finally, we explore the benefits of collaborative localization, where robots in a team assist each other in improving the pose estimation.

Contents

Figures

Tables

Chapter 1

Introduction

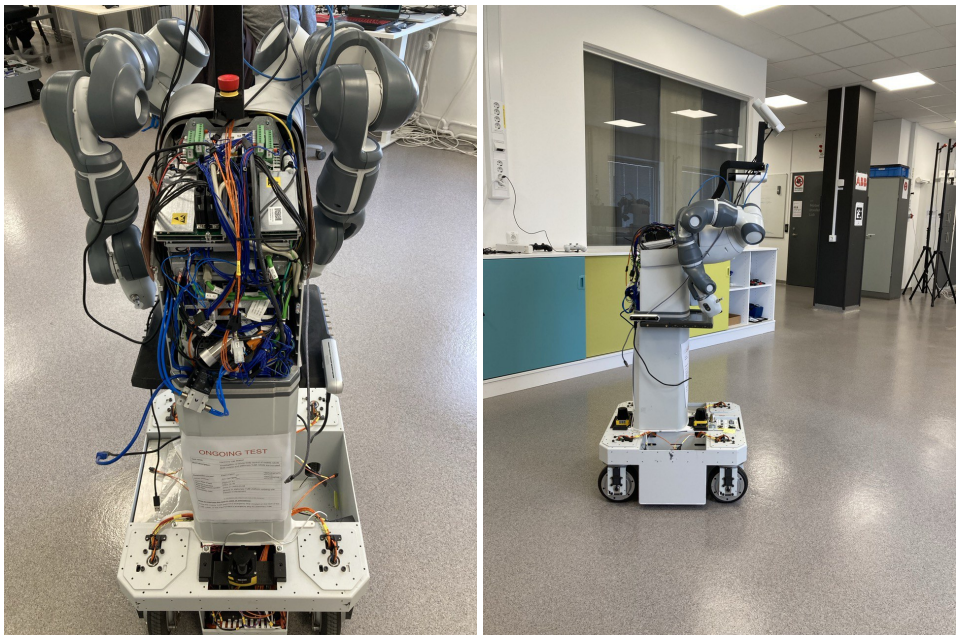
Robot localization is the task of estimating where a robot is located with respect to its environment. Localization is central element in the autonomy of mobile robots, and lies the foundation for more complex tasks such as navigation and planning. When robots are expected to collaborate, with other robots or humans, it is important to have a predefined global frame, such as a map. When localizing using a given map, the estimated poses or the desired goal poses can be shared and understood by other team members. Using a map to navigate in a dynamic, human-occupied environment is a challenging task, due to changes the environment undergoes during long-term operation. Floor plans, which represent static structural elements such as walls, can be utilized to avoid continuous map updates triggered by a changing environment. They are also readily-available when deploying a robot in an new facility. However, the sparse geometric information contained in floor plans can lead to localization failure in the presence of geometric symmetry, as occurs with highly-repetitive environments.

In this doctoral dissertation, we address several scientific challenges, working towards robust long-term localization in changing, human-oriented environments using floor plans. We explore sources of information that can be used to improve on the conventional geometry-based localization approaches. We propose methods of exploiting semantic information for the task of global localization, drawing our inspiration from human navigation, and how humans can efficiently adjust to drastic changes in the environment. We present fusion strategies for integrating information coming from different sensors or perception models. We suggest floor plans as a powerful prior for both long-term localization and mapping. Finally, we verify that our proposed research is also suitable for robotic platforms with constrained resources.

1.1 Motivation

The research was motivated by the Harmony project,[1] which aims to enhance healthcare by introducing robust and safe autonomous robotic mobile manipulation. These mobile robots will operate in human-centered environment, assisting healthcare workers with logistic tasks. This EU-funded project, brought together several universities and industry partners, each contributing to a specific component in the workflow. At that time, I was part of the University of Bonn, and we were tasked with providing the localization and mapping modules, which served

[1] https://cordis.europa.eu/project/id/101017008



(a) (b)

Figure 1.1. The early prototype of the ABB platform for mobile manipulation designed for project Harmony.

as the foundation for more complex robotic tasks, such as navigation, planning, manipulation and human interaction.

According to the scope of the project, we focused our research on indoor global localization. We were also assured that floor-plans would be available for the buildings in which we are expected to operate, therefore we embraced it as a prior. Since we targeted human-oriented environment, we decided to consider textual and semantic cues that are prevalent in such spaces. The robotic prototype was a ground robot (Fig. 1.1), leading us to restrict the localization problem to a 2D plane, or multiple parallel 2D planes in case of multi-story buildings. As the most foundational component in the workflow, we prioritized robustness over accuracy. Additionally, we considered map representations that are easily editable by non-expert humans, at the request of our clinical partners.

1.2 Thesis Overview

The structure of the dissertation is as follows: In the next chapter we provide an overview of the research about single robot localization, with emphasis on long-term localization and the usage of floor plans. In Sec. 3, we describe the engineering efforts that laid the foundation for our scientific research. In the following chapters (Sec. 4 - Sec. 8) we present our research, focusing on long-term localization in floor plans, which was completed and published.

We introduce long-term localization exploiting textual information in Sec. 4, inspired by the usage of textual cues in human navigation. We explore scenarios that contain quasi-static changes, such as closing and opening of door, as well as the highly symmetric environment

of a long corridor. We show that incorporating textual information into a MCL framework significantly increases the robustness. Our work on text-guided MCL has been published in Zimmerman et al. [170].

We extend the work to robust localization using semantic cues in user annotated abstract semantic maps in Sec. 5. With the knowledge that we target human-oriented environments, we again borrow from how humans localize, and exploit spatial relationships between semantic objects to describe places. We provide a user-friendly, easily-adjustable representation for semantic maps that can be hand-annotated and does not require a dedicated mapping procedure. We demonstrate robust localization in floor plans spanning two floors of the IPB lab building, including previously unseen areas. Our work on semantic localization has been published in Zimmerman et al. [171].

We propose an approach for 3D metric-semantic mapping for long-term localization in Sec. 6. Building on the idea of semantic localization, we present an automated procedure for acquiring these semantic maps using 3D object detection from monocular camera, coupled with a complementary localization approach that relies solely on vision sensors. We address the uncertainty of the 3D detection model by learning the characteristic errors, and integrate it into our probabilistic mapping and localization framework. We evaluate our approach on a dataset spanning across 9 months, consisting of dynamic obstacles and structural changes. Our work on 3D metric-semantic mapping has been published in Zimmerman et al. [172].

We address the challenges of localization under resource constraints in Sec. 7. To convince ourselves that semantic localization is a viable solution for robots regardless of their size, we adapt our approach to run onboard a nano-UAV with constrained resources. We optimize and adjust our semantic-guided localization to execute on ultra-low-power processor with miniaturized, short-range sensors and low resolution camera. We establish that semantic localization can be deployed on nano-UAVs and successfully localize in full-scale environments. The content of this chapter covers two publications, Müller et al. [97] and Zimmerman et al. [174].

In Sec. 8 we tackle collaborative localization, in which robots help each other localize by exchanging their belief upon detecting each other. We explore a technique to minimize the computational and communication cost of exchanging and integrating beliefs to improve the localization robustness of a team of robots. We implement and analyze several baselines, including seminal works, and verify the advantage of our proposed approach across multiple environments in simulation and the real world. This work is under review for IROS 2024.

Finally, in Sec. 9 we conclude our work and pinpoint major obstacles in the path of indoor localization.

The main contribution of this dissertation is improving the robustness of long-term indoor localization in human-oriented environments. We achieve that goal through several means:

1. Utilizing readily-available floor plans as a prior for global localization.
2. Integrating semantic and textual cues into a traditional geometry-based localization framework through novel sensor models.
3. Enriching floor plans with semantic information that supports long-term localization, based on analysis of their stability.
4. Leveraging robot collaboration to boost robustness of single-robot localization.
5. Deploying on various robotic platforms, with a wide range of sensing and compute capabilities, to verify our semantic, human-inspired localization paradigm is generally appli-

cable to robots.

Chapter 2

Background

Localization of mobile platforms is a well researched area in robotics [18, 146, 167], and is often classified under state estimation. The state of a robot is the collection of all aspects of the robot and the environment it operates in that can have influence on the future, such as its pose, the configuration of its joints, its velocity or the functionality of its sensors. State estimation aims to recover quantities from sensor data which are not directly observable.

Probabilistic state estimation approaches utilize Bayesian statistics to compute belief distributions, which reflect the robot's internal knowledge about its or the environment's state. Probabilistic methods that estimate the robot's state have proven to be exceptionally robust, and include the extended Kalman filter (EKF) [68], Markov localization by Fox et al. [39] and particle filters often referred to as Monte Carlo localization (MCL) by Dellaert et al. [28]. These seminal works focused on localization using range sensors such as 2D LiDARs and sonars, as well as cameras. For cameras, the global localization task is framed under the visual place recognition framework, for which multiple algorithms have been proposed [7, 27].

The term localization is used to describe three main problems:

- Global localization - an agent needs to infer its pose in a pre-defined global coordinate system, when the initial pose is unknown [39, 28].
- Pose tracking - an agent attempt to correct its odometry drift starting from a known position in a global frame [13, 68].
- Relative positioning - an agent estimates the relative poses of other agents or object in its own coordinate system [54, 81].

Our research focuses on global localization, extending and improving on the Monte Carlo localization framework.

2.1 Monte Carlo Localization

Monte Carlo localization(MCL) [28] is a probabilistic method for estimating a robot's state $\mathbf{x}_t$ given a map m, sensor measurements $\mathbf{z}_t$ and odometry inputs $\mathbf{u}_t$. The MCL algorithm has three main components: the motion model, observation model and resampling, as seen in Fig. 2.1.

As we operate in an indoor environment, the robot's state $\mathbf{x}_t = (x, y, \theta)^\top$ is given by a 2D position and the orientation $\theta \in [0, 2\pi)$. In our case, for a range sensor, an observation

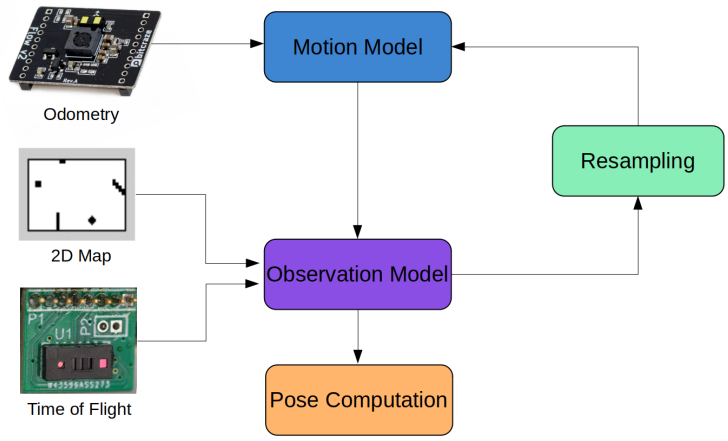


Figure 2.1. The logic flow of the Monte Carlo localization algorithm.

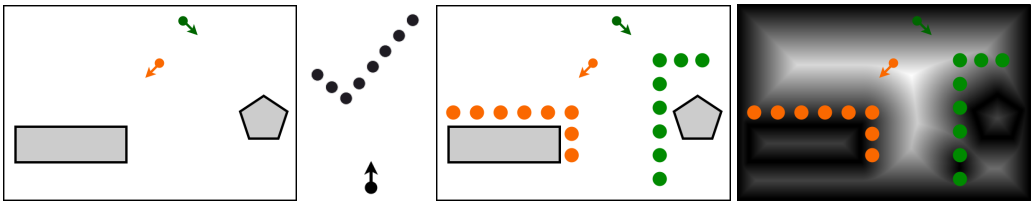


Figure 2.2. Illustration of the Beam-End observation model. (a) An occupancy grid map and two particles representing the hypotheses about the robot's 2D pose. (b) A range measurement composed of 8 beam end points in the robot's coordinate system. (c) The measurement transformed to the global frame based on each particle's pose hypothesis. (d) The EDT of the map, overlaid with the positions of the transformed scans.

$\mathbf{z}$ is composed of K individual range measurements(beams) z^k and the map m is represented by an occupancy grid map [94]. Occupancy grid maps are a discrete representation of the environment, where each cell in the grid is a random variable corresponding to whether the cell is an occupied or a free space.

We use a particle filter to represent the belief about the robot's state $p(\mathbf{x}_t \mid \mathbf{z}_t, m)$, where each particle $s_t^i = \left(\mathbf{x}_t^i, w_t^i\right)$ is represented by a state $\mathbf{x}_t^i$ and a weight w_t^i. The proposal distribution $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t)$ is sampled when a new motion prior $\mathbf{u}_t$ is available, by applying the prediction step

$$p\left(\mathbf{x}_t \mid \mathbf{z}_{t-1}\right) = \int p\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t\right) p\left(\mathbf{x}_{t-1} \mid \mathbf{z}_{t-1}\right) d\mathbf{x}_{t-1}, \tag{2.1}$$

where we assume the current state $\mathbf{x}_t$ only depends on the previous state $\mathbf{x}_{t-1}$ and the odometry input $\mathbf{u}_t$. In the second step we update the prior distribution by applying the observation model

$p(\mathbf{z}_t \mid \mathbf{x}_t, m)$, which incorporates information obtained by the sensors into the state estimation

$$p(\mathbf{x}_t \mid \mathbf{z}_t, m) = \frac{p(\mathbf{z}_t \mid \mathbf{x}_t, m)\, p(\mathbf{x}_t \mid \mathbf{z}_{t-1})}{p(\mathbf{z}_t \mid \mathbf{z}_{t-1})}. \tag{2.2}$$

For every observation, each particle is weighted according to the likelihood of the observation given its state, i.e., $w_t = p(\mathbf{z}_t \mid \mathbf{x}_t, m)$. After assigning to each particle a new weight based on the observation, we resample the particles. In the resampling step, particles with high weight are likely to be sampled and particles with low weight are discarded. Repeating this process allows us to focus on the strong hypotheses, until the particle filter converges to a single pose estimation. Even in cases where multiple hypotheses exist due to geometric symmetries, eventually the filter will collapse to one mode due to sensor and odometry noise, and issues arising from float precision.

In our implementation of a motion model for holonomic robots, we decompose the motion prior $\mathbf{u}_t$ into three components, forward, sideways and rotation, $\mathbf{u}_t = (f, s, r)$. The odometry noise $\boldsymbol{\sigma}_{\text{odom}} \in \mathbb{R}^3$ is assumed to be Gaussian and is applied separately to each component of the odometry

$$\tilde{\mathbf{u}}_t = \begin{bmatrix} \tilde{f} \\ \tilde{s} \\ \tilde{r} \end{bmatrix} = \begin{bmatrix} \mathcal{N}(f, \sigma_{\text{odom,f}}) \\ \mathcal{N}(s, \sigma_{\text{odom,s}}) \\ \mathcal{N}(r, \sigma_{\text{odom,r}}) \end{bmatrix}. \tag{2.3}$$

As described in Eq. (2.1), the particle's pose $\mathbf{x}_t$ is updated based on the noised motion prior $\tilde{\mathbf{u}}_t = (\tilde{f}, \tilde{s}, \tilde{r})$ and the previous state $\mathbf{x}_{t-1}$

$$\begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} = \begin{bmatrix} x_{t-1} + \tilde{f}\cos\theta_{t-1} - \tilde{s}\sin\theta_{t-1} \\ y_{t-1} + \tilde{f}\sin\theta_{t-1} + \tilde{s}\cos\theta_{t-1} \\ \tilde{r} + \theta_{t-1} \end{bmatrix}. \tag{2.4}$$

For range-based localization, a common choice of an observation model for the update step (Eq. (2.2)) is the beam-end model [146]

$$p(z_t^k \mid \mathbf{x}_t, m) = \frac{1}{\sqrt{2\pi\sigma_{\text{obs}}}} \exp\left(-\frac{\text{EDT}(\hat{z}_t^k)^2}{2\sigma_{\text{obs}}^2}\right), \tag{2.5}$$

where $\hat{z}_t^k$ is the end point of the beam in the map m, and EDT is the Euclidean distance transform [37] that indicates the distance to an occupied cell in the occupancy map. σ_{obs} is the measurement noise, reflecting inaccuracies in the sensor. The procedure is illustrated in Fig. 2.2. First, we start with an occupancy grid map m, a particle's pose $\mathbf{x}_t^i$ and an range measurement $\mathbf{z}_t$, where z_t^k corresponds to a single range measurement annotated at a green point. A range measurement z_t^k is a point relative to the robot's pose, and given a particle's pose $\mathbf{x}_t^i$, we convert it to an absolute pose $\hat{z}_t^k$ in the global coordinate frame. We then use the position of the transformed point $\hat{z}_t^k$ to sample the EDT. As expected from range sensors, this end point should correspond to an occupied cell in the grid map, which is a equivalent to a low-valued cell in the EDT.

Eq. (2.5) computes the weight of a particle given a single beam z_t^k, therefore we must consider how to compute the weight for the full observation $\mathbf{z}_t$. The product of likelihood model assumes scan points are independent of each other. With the high angular resolution of modern LiDARs this assumption does not hold. To address the overconfidence problem of the product

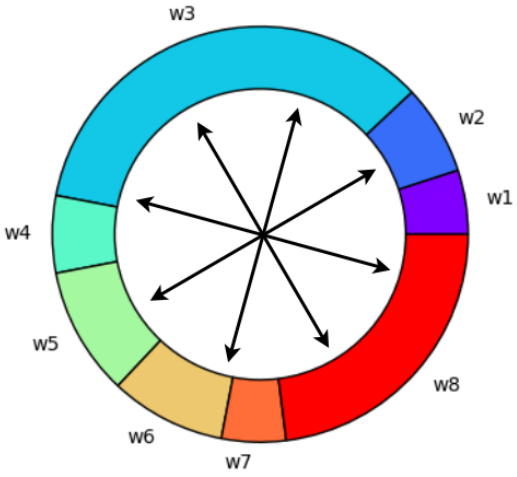


Figure 2.3. Visualization of the low-variance resampling method, where particles are sampled at fixed intervals based on their accumulated weight. Particles with high weight, such as w_3 are sampled multiple times, while particles with low weight, i.e. w_1, are more likely to be skipped.

of likelihood model, we decided to use the product of experts model [90], where the weight of each particle is computed as the geometric mean of all scan points

$$p(\mathbf{z}_t \mid \mathbf{x}_t, m) = \prod_{k=0}^{K} p(z_t^k \mid \mathbf{x}_t, m)^{\frac{1}{K}}. \tag{2.6}$$

For the resampling, we use low-variance resampling [146], which is visualized in Fig. 2.3. For N particles, we sample at a fixed interval $\frac{1}{N}$ based on the accumulated weight $w^i_{\text{acc}} = \sum_{i=1}^{N} w^i$. To estimate the robot's pose, we compute the weighted average of all particles in the filter.

2.2 Long-Term Localization

A problem arises when dynamic objects are detected in the scans and observations cannot be correctly matched to a given map. Outlier rejection can be utilized for handling dynamics in the context of state estimation, such as robust kernels proposed by Chebrolu et al. [21]. Thrun et al. [146] also incorporate the appearance of unexpected objects in the sensors model, by modeling it as a component of the sensor noise. Dynamic objects would result in measurements that are shorter than the true range, which can be determined using ray casting. They point out that the likelihood of sensing unexpected objects decreases with range, and can be described by an exponential distribution. Another aspect of scene dynamics is changes that are longer-lasting and not as fast to appear and disappear like moving objects. These long-lasting changes can be closing and opening of passages, transferring large packages from one place to another and shifting of large furniture. Since those changes are more constant, standard filtering techniques will fail to remove them.

Stachniss and Burgard [141] address the case of slow-dynamic environments, specifically the case of closing and opening doors. They first learn the different environmental configuration by observing the environment at different times and clustering sub-maps that represent possible environmental states. They extend the standard MCL algorithm, by also considering the configuration of the environment as opposed to using a static map.

Several localization works focus on the challenge of seasonal changes in an environment [151, 88]. However, these methods require a sequence of images to localize, by matching them against a database of pre-recorded sequences. A different approach by Krajník et al. [65] tries to capture periodic changes by representing every cell in the occupancy map as a periodic function. This approach to life-long mapping and localization is based on the assumption that changes to the environment are caused by processes which are (pseudo-) periodic. In standard occupancy grid maps, each cell stores a single value corresponding to its probability of it being occupied. With frequency-enhanced maps, each cell stores a probabilistic function of time composed of several harmonic components. To estimate these parameters, long-term observations are analyzed using non-uniform Fourier transform. However, these methods require a sequence of images to localize, by matching them against a database of pre-recorded sequences. In our work, we do not assume to have prior knowledge on changes that may occur in the map, nor do we require long sequences of images to match against.

To tackle more general semi-permanent changes, Valencia et al. [148] suggest using multiple static maps, each corresponding to a different time scale. They localize in NDT-occupancy grid maps [129], where each cell holds a occupancy probability function, resulting in a map that is piece-wise continuous instead of discrete. NDT-MCL handles quasi-static changes by constructing short-term maps and alternating between localization with a known map, and mapping with a known pose. Biber and Duckett [10] extend the concept of a map from a static, geometric description to a spatio-temporal representation, inspired by the human memory. They emphasize the plasticity-stability trade-off, between updating the map based on new observations and preserving the old ones. They define a timescale parameter that determines the decay rate for old measurements, and allows to classify observations as outliers, while maintaining several maps over multiple timescales. They track both stationary and non-stationary elements of the environment using robust statistics.

Sun et al. [143] account for dynamic obstacles by using a distance filter, such that observations that are closer than the closest obstacle in the map are rejected as outliers during localization. Localization is implemented using a particle filter with additional scan matching to improve the accuracy. The scan matching also provides a score which is used as confidence measure, which can help in deciding whether to update the map. The occupancy grid map is dynamic and is modeled by an HMM. Similarly, in the work of Tipaldi et al. [147], a Hidden Markov model [5] is assigned to every grid cell, creating dynamics occupancy grids that can be updated. The state transition probabilities are computed using the EM algorithm, and the localization is implemented using the a Rao-Blackwellized particle filter (RBPF)[46]. These methods require continuous update of the map, while we handle changes without altering the map.

2.3 Localization in Floor Plans

Localization in feature-rich maps, often constructed by range sensors, is well-established [93]. However, there are advantages for using sparse maps such as floor plans for localization. We consider floor plans to be sparse as they lack much of the detailed geometric knowledge contained in sensor-based maps, and can be shortly summarized as a list of geometric primitives rather than the explicit occupancy grid maps [93]. Floor plans are often available for buildings and do not require prior mapping with LiDARs or other range sensors. Their sparsity also means they do not need to be updated as frequently as detailed maps that include possibly mov-

ing objects, such as furniture. Their downside is their lack of details, which can render global localization challenging when faced with multiple identical rooms. Another issue is a possible discrepancy between the plans and the construction [13].

Boniardi et al. [13] exploit floor plans to represent the immutable features of a building, as their methods switches between pure localization and pose graph estimation. In their work, Boniardi et al. [13] show robustness to structural changes, but do not address the challenge of dynamic obstacles.

Li et al. [72] tackle the issue of improper maps - from inaccurate floor plans to hand-drawn maps, by introducing a new state variable, scale, to address the scale difference between the map and the observed structures. For every step of the localization, they look at local sub-map, and the measured LiDAR ranges are associated with the nearest occupied cells in the sub-map. A cost function is then minimized to infer the state variables, (x, y, θ, s), using stochastic gradient descent [126].

Watanabe et al. [157] also address the discrepancies between floor plans and the constructed buildings, as well as the presence of dynamic objects. By extracting planes from 3D LiDAR scans, they filter outliers, and ensure the scan contains only the points derived from walls and columns. They estimate the transformation between the scan and the floor plan using generalized ICP [133] framework, which can also overcome inconsistencies between the floor plan and the real structures.

Winterhalter et al. [160] acquire full 6 DoF pose estimation in floor plans, combining RGB-D images and highly accurate visual-inertial odometry. Their approach is based on MCL, optimized by KLD sampling [38], a technique which adjusts the number of particles by bounding the approximation error introduced by the sample-based representation of the particle filter. They apply the Beam-End model [146] by randomly sampling K range measurements from the dense depth images. They model objects that are not present in the floor plan by uniform distributions, and experimentally estimate the parameters by evaluating the measurement error on posed depth images collected in their office environment. All approaches enable tracking but not global localization.

Maffei et al. [79] compute the Free-Space Density(FSD) [78] of floor plan to globally localize using RGB-D images. Similarly to the EDT, the FSD of a given position in space is a measure of the free-space surrounding such position, which is computed using a circular kernel. They propose a method of localizing in an incomplete local map by estimating an interval of possible FSD values.

Wang et al. [154] annotate floor plan corners as landmarks, and match them against corners extracted from a 3D LiDAR scan. They present a factor graph-based localization approach that considers the uncertainty of the landmarks, and a robust data association method to match the detected corners with the annotated corners in the floor plan. As they only consider long vertical edge features extracted from the 3D LiDAR scan, they can filter dynamic objects and some of the permanent objects that are excluded from the floor plan, such as furniture.

Both Boniardi et al. [14] and Howard-Jenkins and Prisacariu [55] align room layout inferred from an image with the floor plan of a scene to globally localize. While Boniardi et al. [14] make assumptions on the scene's geometric to extend the 2D floor plan to a 3D model, Howard-Jenkins and Prisacariu [55] learn a general prior over 2D floor plans, implicitly hallucinating 3D structure across the entire plan. While these works address the difficulties of localizing in floor plans in different ways, their exclusive reliance on geometric information is not sufficient for global localization in a highly repetitive indoor environments.

Chapter 3

Robots and Infrastructure

The goal of robotics research is to design, assemble and evaluate intelligent systems than can assist humans in a variety of tasks. The field of robotics is applicative and practical, and success is measured when robotic systems display the desired behavior in the wild. Knowing robotics is knowing your robots, and working with robots is essential to fulfilling the purpose of a PhD - becoming an expert in the field.

While for localization the scientific contribution is mostly algorithmic in nature, such research cannot be conducted without robots. The approaches proposed in the following chapters could not be properly developed and evaluated in the absence of real robots - from collecting custom datasets to onboard deployment and real-life testing.

In this chapter, we present the engineering and infrastructure work that served as the foundation of our scientific research. We present an infrastructure for accurate global localization, providing ground truth poses with which we evaluated our methods. We detail the trial-and-error involved in developing a robotic platform, a critical part in enabling the autonomy of robots in real-life environments. We introduce the sensor setups that were used in our work (Fig. 3.12), and explain the calibration procedure required to describe all sensors in the same coordinate system.

3.1 Global Localization Infrastructure

Accurate, dense ground truth pose is essential for many robotics research domains. Naturally, when developing localization algorithms, ground truth poses are crucial for evaluation. However, these poses are also useful for navigation, planning and manipulation tasks. The use of salient visual features for localization has been long established [3, 18, 162]. A gold standard for robot localization is the AprilTag [104], a visual fiducial system. The target tags are open-sourced[1] and easy to manufacture, with near-optimal fiducial design and coding system [17, 104]. The detection software for the target is also open-sourced[2], and is sufficiently robust to changing lighting conditions and a wide range of viewing angles. The AprilTag detector detects the 2D image coordinates of the marker's corners, as well as the marker's ID and a Hamming error. Assuming a calibrated camera, this allows us to compute the marker's precise 3D pose relative to the camera via spatial resection [139, 130], also known as the PnP

[1] https://github.com/AprilRobotics/apriltag-imgs
[2] https://github.com/AprilRobotics/apriltag

algorithm [61, 69]. Spatial resection in photogrammetry refers to estimating the six degrees of freedom, $(x, y, z, \theta, \phi, \psi)$, encoding the 3D pose of a single image, based on known 3D control points. These parameters are computed by finding the correspondence between the known 3D control points and their respective 2D locations in the image. The size of AprilTag target is known prior to installation, and therefore can be used to set the 3D control points. When the 3D coordinates of the targets are known, we can estimate the absolute pose of the camera in our global coordinate system.

3.1.1 Localization infrastructure - IPB Lab

For the research conducted in the university of Bonn, a global localization infrastructure was installed in the first and second floor of the IPB lab. The AprilTags were assembled manually, and installed in a grid-like structure across the ceilings of all rooms. The markers were aligned precisely by projecting a laser beam across the ceiling and placed in fixed intervals, as shown in Fig. 3.1. The markers were densely placed, at around 1 tag/m^2, with more than 300 tags covering areas of interest in the lab.

To extract the 3D coordinates of the AprilTag markers, we scanned the lab with a high-precision terrestrial laser scanner (TLS). Using the ICP algorithm [1], we combined over 40 scans into a large pointcloud of 1.6 billion points. Once all markers were present in the same coordinate system, we selected a slice of the pointcloud that contained only the ceiling. Since the TLS provides extremely dense pointclouds, it was possible to detect the AprilTag markers from the pointcloud by using orthographic projection. In orthographic projection, a 3D object is projected to 2D without perspective distortion, so parallel lines are preserved. Therefore, we were able to to automatically detect the 3D coordinates for the markers in a unified frame.

A wide-angle up-looking camera was chosen for detecting the tags, capturing multiple tags in every frame (Fig. 3.2). The AprilTag detection code was wrapped in a ROS node, and while the ground truth pose estimation was mostly an offline procedure, it can also run onboard our ground robots at 10 Hz. Once the 2D corners and marker IDs were detected in an image, spatial resection was implemented using the BACS library [131]. While popular libraries like OpenCV [15] include implementations of spatial resection for pinhole model cameras, BACS also offers support for fisheye (or equi-distant) camera models and provides the covariance matrix for the pose. Since multiple markers were detected at every frame, a least squares optimization was performed to compute the 3D pose. With our latest robotic platform, the upgraded Dingo (Sec. 3.2.2), we achieved a sub-centimeter accuracy for the the ground truth pose estimation.

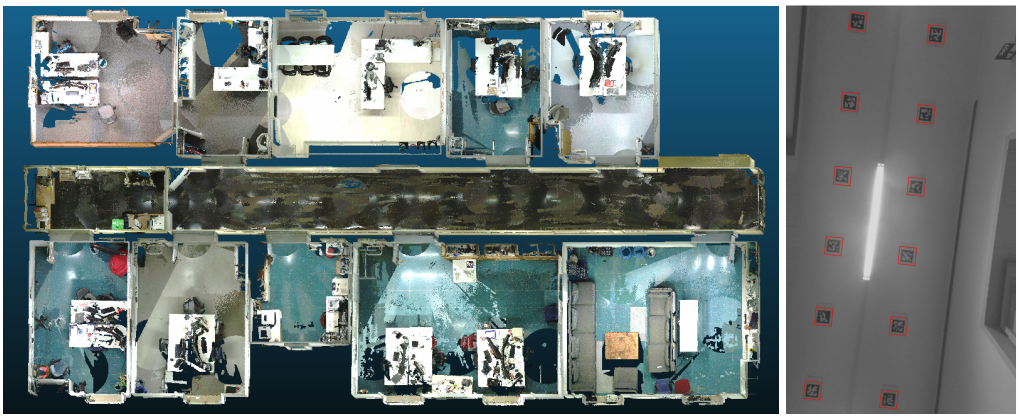
3.1.2 Localization infrastructure - PBL Lab

For the research conducted in ETH-Zurich, a global localization infrastructure was installed in the PBL lab. Due to hardware related constraints, it was not possible to have a dedicated onboard camera for ground truth pose extraction. In this case, the front-facing camera was utilized also for ground truth pose estimation, and the markers were placed on the walls instead of the ceiling.

148 tags were manufactured and installed in an area of 280 m^2. Since the camera had a limited FoV of 60^o, we decided to place the markers in a 2×2 grid, to increase the probabilities of detecting multiple tags in a single frame. Examples of the AprilTag markers detected from the onboard low-resolution camera can be seen in Fig. 3.3.

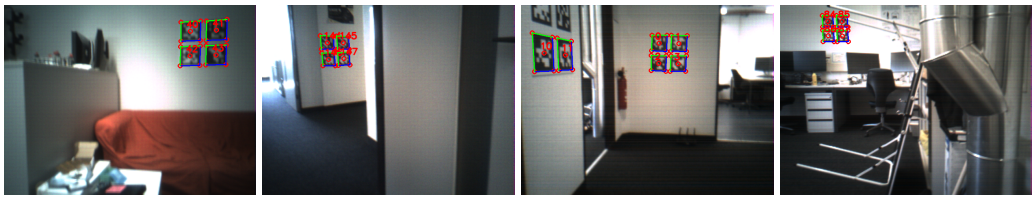
(a) (b) (c) (d)

Figure 3.1. The making and installation of AprilTags in IPB lab. (a) Hand-made AprilTag marker. (b) Laser pointer beam used to align the markers. (c) Geodetic tripod with a level. (d) Installing the markers along the IPB lab's corridor.



(a) (b)

Figure 3.2. The localization infrastructure in IPB lab. (a) A top-view of the sub-sampled pointcloud constructed from combined scans. (b) Multiple detected AprilTag markers in a frame captured by the up-facing wide-angle camera.



(a) (b) (c) (d)

Figure 3.3. Multiple AprilTag markers detected in each frame in the PBL lab.

(a) (b) (c)

Figure 3.4. The localization infrastructure in PBL lab. (a) Verification against IR tracking system (Vicon) for the pose estimation pipeline. (b) Specialized training in the Hilti facility. (c) The unified pointcloud from the PBL lab scans.

Similarly to the procedure performed in the university of Bonn, we scanned the lab with a high-precision terrestrial laser scanner(TLS) provided by Hilti, and 18 scans were merged using the ICP algorithm. In order to correctly operate the scanner we underwent training in the Hilti facility in Schaan.

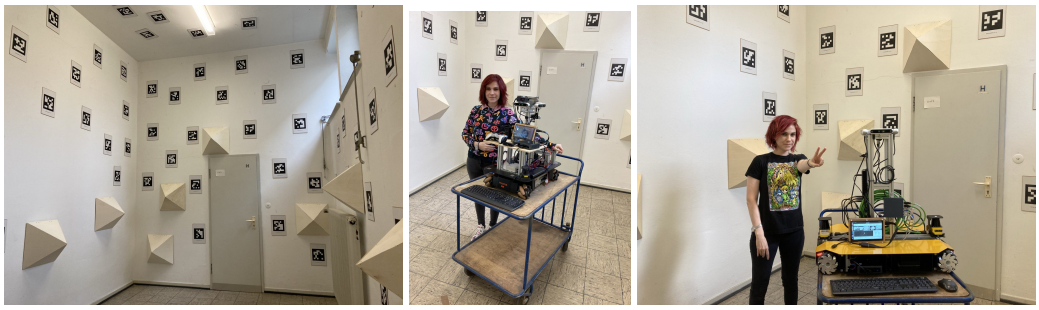
The extraction of 3D coordinated from the combined pointcloud was slightly more involved than the one in IPB lab, as the markers were placed on the walls and not on the ceiling. Therefore, we were not able to cleanly and automatically extract surfaces containing the AprilTags.

Since the front-facing camera was not a wide-angle camera, we were able to utilize the open-source OpenCV implementation of the PnP algorithm. To verify the correctness of our pipeline, we combined AprilTag markers with IR markers in a lab equipped with a motion capture system (Vicon) and compared our pose estimation to the ground truth pose provided by the tracking system. For the various test points we considered, the errors were around 0.05m.

3.1.3 Calibration infrastructure - IPB Lab

To accurately combine information from different sources, the transformation between the sensors must be known. We refer to the process of find these transformations as sensor calibration. As we have multiple sensors on our robot platforms (Fig. 3.12), we had to establish a procedure to efficiently and accurately calibrate our sensors.

The calibration room (Fig. 3.5) of IPB lab was established for that purpose. AprilTag markers were installed to aid with calibration of cameras and 3D pyramids were mounted on the walls to assist with LiDAR calibration. The room was scanned with TLS and the 3D coordinates of the AprilTag markers were extracted. First, the robot is placed in different locations in the room, while recording from all sensors. Then the sensors are synchronized such that each time stamp includes a measurement from every sensor. A sequence of multiple synchronized frames is then matched against the known 3D model of the room, and the transformations between sensors are extracted in a least-squares fashion. Calibration of the intrinsic camera parameters is performed prior to this. A detailed description of the calibration process is provided by Wiesmann et al. [159].



(a) (b) (c)

Figure 3.5. (a) The calibration room in the IPB lab. (b) Calibration of the YouBot. (c) Calibration of the Dingo.

3.2 Robots

3.2.1 Kuka YouBot

The Kuka YouBot is a small, omnidirectional mobile robotic platform designed for scientific research and education. Initially, the YouBot in the IPB lab was equipped with an outdated depth camera and a pair of 2D Hokuyo UTM-30LX LiDAR sensors. The drivers were installed on a laptop with Ubuntu 16.04. To comply with the sensor setup of the Harmony project, the vision sensors were upgraded to include 4 sideways-looking Intel RealSense RGB-D (D455) with a 360^o FoV and a Kinect 2 sensor. Additionally, an up-looking GoPro 5 camera was mounted to extract the ground truth poses via the AprilTag infrastructure. The GoPro 5 was chosen for its wide-view and self-stabilizing properties.

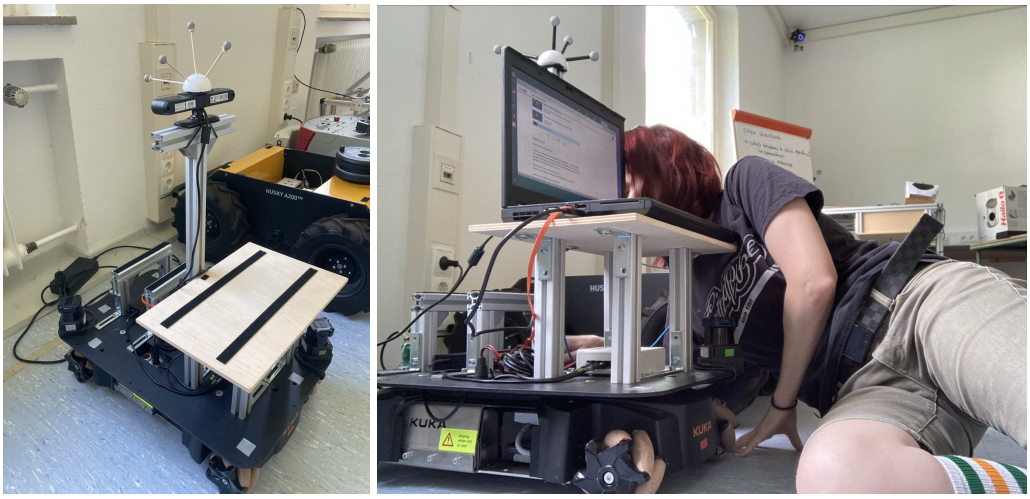
However, the Lenovo laptop could not handle the large bandwidth of streaming from all cameras at 30 Hz, and placing the laptop onboard increased the instability of the sensor mount. The Intel NUC10i7FNK was chosen as an onboard computer, and was powered from the YouBot battery using a DC converter. While the NUC could handle the USB bandwidth for camera streaming, it could not provide power to all connected cameras. Therefore a powered-USB hub was added to the setup, also supplied by the robot's battery. The official support for the YouBot drivers ended with Ubunntu 16.04, but the drivers were adjusted, built from source and upgraded to Ubuntu 20.04 and ROS Noetic.

The YouBot served as our data collection platform, contributing to several research works, and also functioned as testing platform for Harmony-related tasks. Since Kuka stopped manufacturing the YouBot, it was difficult to find replacement parts, and custom batteries were self-made in the university of Bonn. The IPB YouBot began to degrade after 8 years of operation, and it was time to design the new generation of indoor IPB robots.

3.2.2 Clearpath Robotics Dingo

The Clearpath Robotics Dingo is a small, omnidirectional mobile platform for indoor operation. We chose it to replace the aging YouBot, for its size, cost and form, which allows to mount a variety of sensors.

The Dingo was purchased in its most basic configuration - a chassis and an MCU, as shown in



(a) (b)

Figure 3.6. The initial configuration of the Kuka Youbot platform.

(a) (b) (c) (d)

Figure 3.7. Incremental improvement of the sensor setup of the YouBot. (a) 3D printing of a sensor mount. (b) An intermediate setup with the Intel RealSense D435 and Intel T265 supported by the 3D printed mount. (c-d) The final camera setup with 4 Intel RealSense D455, Intel RealSense T265, Kinect Azure and the up-looking GoPro 5 camera.

(a) (b)

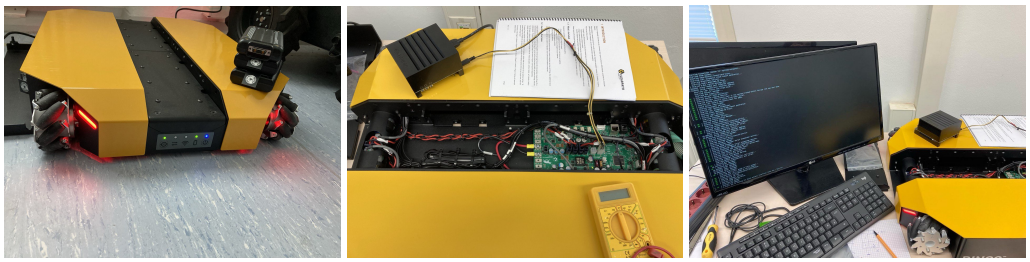
Figure 3.8. Transitioning from laptop dependency to onboard computation using an Intel NUC.

Fig. 3.9. The official recommendation for a compute platform was the Nvidia Jetson, which was purchased separately. At the time, drivers were supported up to Ubuntu 18.04/ROS Melodic, and they were installed on a Nvidia Jetson Xavier AGX, which was powered from the robot's battery.

Preliminary testing with the Jetson revealed that it could not handle the large bandwidth from all cameras. Therefore, we decided to use the Intel NUC again, but no drivers were provided for this platform. We decided to install Ubuntu 20.04 and ROS Noetic on the NUC, and again compiled all the drivers from source. This update was partially successful as we could control the robot with a joystick, but the indicator lights were blinking rapidly, signaling a battery issue. We investigated the matter and found that the battery status messages format had changed between the supported ROS version and the version we built from source. To remedy this, the firmware on the Dingo's MCU was updated, a convoluted procedure that was remotely assisted by Clearpath Robotics engineers.

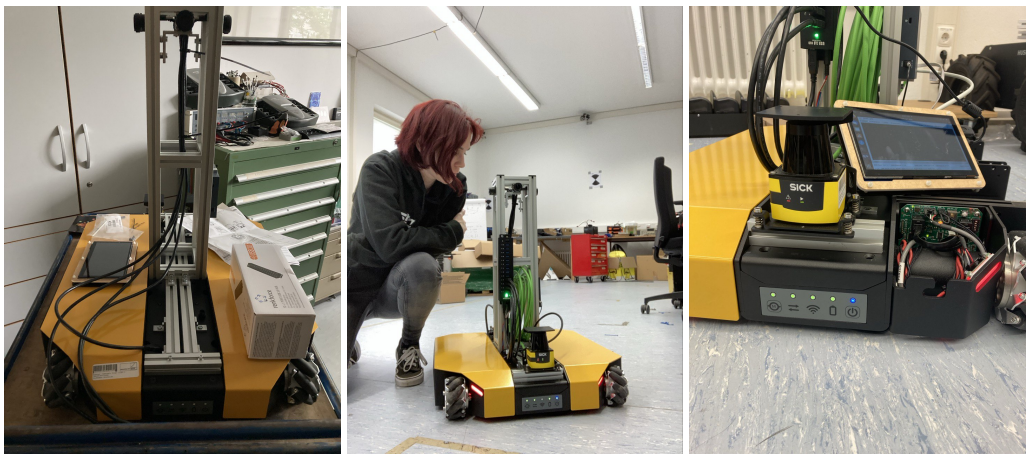
With project Harmony in mind, the sensor mounting for the Dingo was designed. Learning from the mistakes made with the YouBot, we improved the structure holding the cameras to offer more stability (Fig. 3.10). To reduce the USB bandwidth, we chose an Ethernet-based BlackFly S camera for our ground truth extraction. The BlackFly, combined with a wide-angle lense, proved easier to calibrate than the YouBot's GoPro. Additionally, 4 sideways-looking Intel RealSense RGB-D (D455) with a joint 360^o FoV were mounted, and a pair of SICK TIM781S LiDAR sensors.

The Intel NUC proved sufficient for data collection, but for testing localization algorithms onboard we also made use of the Nvidia Jetson. All sensors and compute platforms were powered by the board (Fig. 3.11), and required thoughtful wiring work to accomplish.



(a) (b) (c)

Figure 3.9. The early phases of the Dingo platform. (a) The dingo platform upon arrival, with no sensors or compute platform. (b) powering the Nvidia Jetson from the Dingo battery (c) First installation of the Dingo drivers on the Jetson.



(a) (b) (c)

Figure 3.10. Sensor suite design for the Dingo. (a) Mechanical work for sensor mounting. (b) Installing the LiDAR sensors. (c) Running all drivers onboard.

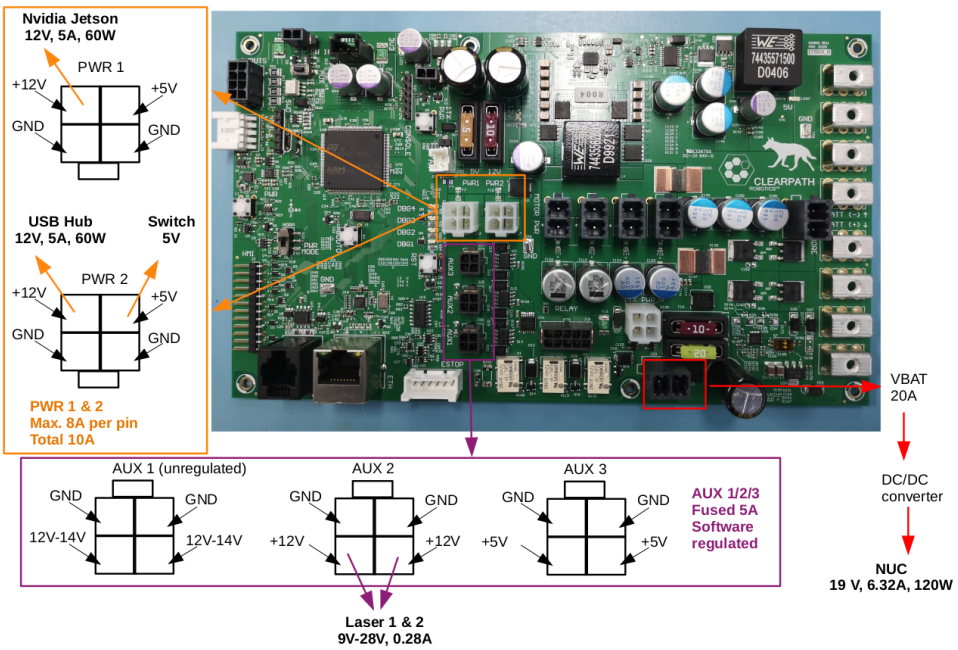


Figure 3.11. Electricity diagram for the Dingo.

(a) (b)

Figure 3.12. The final setup of our ground robots, with 2D LiDAR scanners and 4 cameras providing 360° coverage. The up-ward facing camera is only used for generating the ground truth via AprilTag detections. (a) YouBot. (b) Dingo.

Chapter 4

Textual Information for Robust Localization

Localization in a given map is a fundamental capability required by most autonomous robots operating in indoor environments, such as offices or hospitals. These environments are often populated by people, also undergoing “quasi-static” changes such as closing of doors, objects temporarily misplaced, or moved furniture that is not reflected in the given map that was recorded at a different point in time. Such changes, which we refer to as “quasi-static” in contrast to dynamic ones such as moving people, result in sensor observations that substantially differ from the map and can lead to localization failure, as illustrated in Fig. 4.1, where closed doors in a corridor remove localization cues, which can then lead to ambiguities.

To overcome such localization challenges, readily available sources of information can be exploited to aid pose estimation. One example is using WiFi signal strength [57] from existing access points to aid the localization. Another example is using textual information that is part of the building infrastructure. Textual cues are often used by humans to navigate in the environment and are therefore available in most buildings designed for humans. With the recent advances in deep learning-based text recognition [137], we can reliably and efficiently decode textual content from images and utilize these hints in our localization approach. Surprisingly, there exist only a few approaches [26, 117] in the robotics community to exploit text spotting or optical character recognition (OCR) for robot localization. The main contribution of this work is a localization framework that integrates text spotting into a particle filter to improve localization. To this end, we build maps indicating the likelihood of detecting room numbers across the environment, under the assumption that room numbers are rarely change. The locations with high likelihood for successful detection are then used to inject particles when a known sign is detected. The textual cues allow us to globally localize with a small number of particles enabling online performance on mobile robots with limited computational resources. In our experiments, we show that our approach is able to (i) localize in quasi-static environments, (ii) localize in an environment with low dynamics, (iii) localize in different maps types – a featureless floor plan-like map, and LiDAR-based, feature rich map. Furthermore, our approach runs online on an onboard computer.

This work, titled "**Robust Onboard Localization in Changing Environments Exploiting Text Spotting**" [170], was accepted to *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2022.

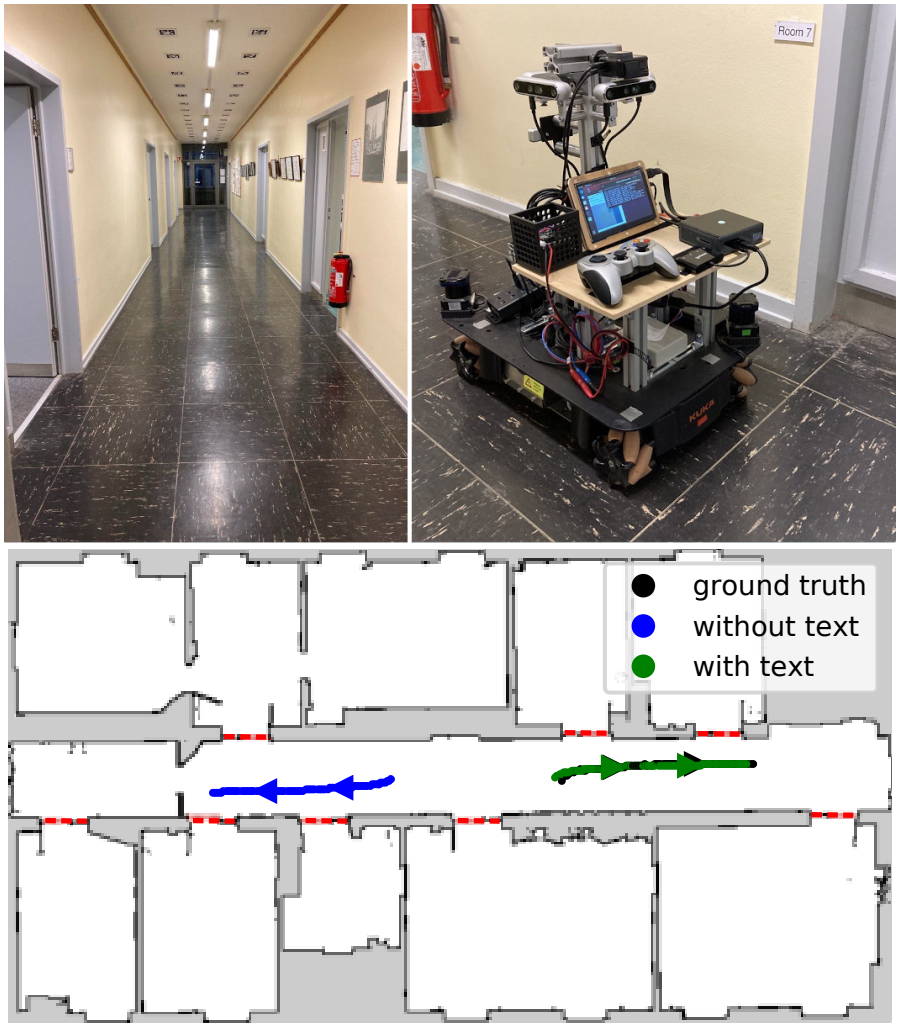


Figure 4.1. Top Left: The corridor in which the experiment took place in. Top right: The Kuka YouBot platform that was used for data collection, equipped with 2D LiDAR scanners and cameras that cover the complete 360° field-of-view we utilize for text spotting. Bottom: The results of of localization in a corridor with closed doors (indicated by red lines), which are not reflected in the map, with and without textual cues.

4.1 Related Work

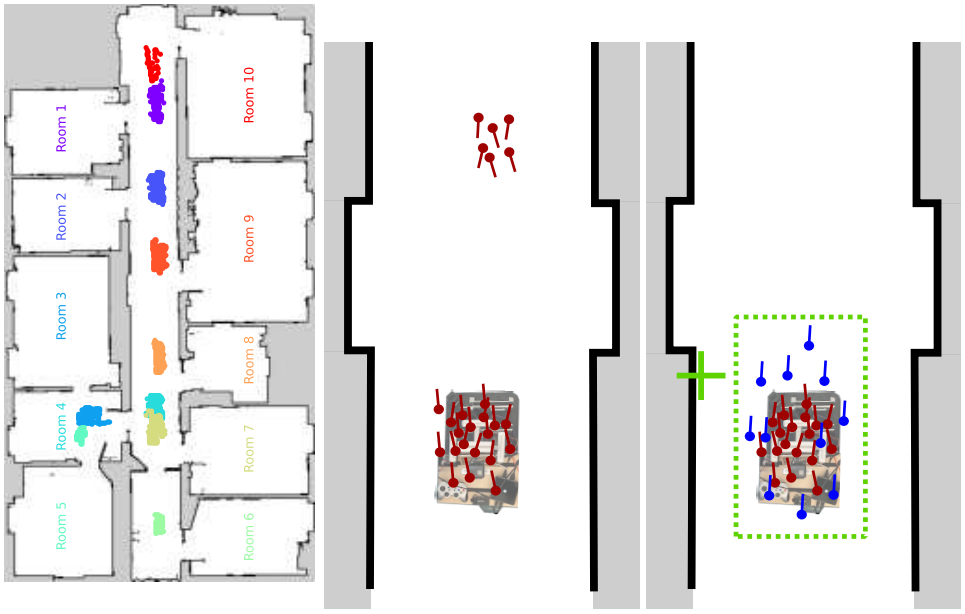
The problem of localization using CAD floor plans has been seldom addressed in literature, and an overview of the approaches is given in Sec. 1. To assist global localization, additional modalities were considered.

Fingerprinting, a common indoor positioning technology based on signal strength data, is also applicable for robot localization. Xu et al. [164] propose acoustic-based wide-area localization in floor plans utilizing Bluetooth measurements and the built-in MEMS sensors of a smartphone. However, they rely on an external infrastructure of acoustic anchors and their localization accuracy is insufficient. Miyagusuku et al. [89] explore ways in which different WiFi access points can be combined in a joint distribution, given that every access point has its own likelihood map constructed as function of the signal strength. Ito et al. [57] use WiFi signal strength to estimate the initial pose, based on signal strength maps that were previously constructed, and then track the pose in a floor plan using RGB-D images. Hahnel et al. [49] present an approach for both mapping and localization based on RFID technology. They provide a probabilistic model for mapping RFID tags into occupancy grid maps, and then show how the RFID detection can improve upon LiDAR-based localization. Joho et al. [60] suggest a sensor model for RFID that combines the likelihood of detecting a tag at a given pose and the likelihood of receiving a specific signal strength. We take inspiration from these papers for building our text likelihoods/priors but apply it for a different modality. Unlike these approaches, our text-augmented floor plans can be also constructed by hand, and does not require a dedicated mapping process.

Considerable amount of information is helping humans navigate, from publicly available maps to direction signs. Vysotska and Stachniss [150] use publicly available maps, like Open Street Map, to localize with LiDAR in outdoor environments. While humans rely often on textual cues to navigate, exploitation of text for localization is not commonly explored. It was suggested by Radwan et al. [117] but considers outdoor environment and usage of Google Maps, while our approach tackles indoor environments. Another implementation of text spotting in a MCL framework is presented by Cui et al. [26], who rely on text detection as its only sensor model. This differs from our work, which uses a 2D LiDAR-based sensor model and only leveraged text to improve global localization. The advantage of our method is that we are able to localize even in the absence of textual cues. Furthermore, in the work of Cui et al. [26], text spotting is trained specifically for spotting parking space numbers, while we use a generic, off-the-shelf text spotting that performs well on a variety of textual cues [137].

4.2 Approach

Our goal is to globally localize in an indoor environment that can undergo significant structural changes using 2D LiDAR scanners, cameras and wheel odometry. In sum, we achieve this by building upon the Monte Carlo localization framework. To aid with global localization and recover from localization failures, we use a text spotting approach inferred from camera images to detect room numbers of an human oriented environment. To integrate the textual cues, we create text likelihood maps, which indicates the likelihood of detection of each room number as a function of the robot position. We inject particles corresponding to the locations suggested by the text likelihood.



(a) (b) (c)

Figure 4.2. Particle injection with text spotting. (a) The text likelihood maps, based on the collected data, indicate the locations in which detection of each room number is likely. The likelihood maps are used for particle injection when a detection of a known text cues occurs. (b) Before detection, we have a situation with multi-modal distribution of particles (shown in red) as the corridor with closed doors is a symmetric situation that cannot be resolved just using the LiDAR scans. (c) With the first text detection (indicated by the green cross), we can inject new particles inside the bounding box extracted from the text map. We replace low weighted particles by new particles (shown in blue) that are uniformly distributed inside the corresponding bounding box of the text detection (shown by a dashed green line).

4.2.1 Text Spotting

Text spotting can traditionally be split into text detection, i.e., localizing a bounding box that includes text, and text recognition, i.e., decoding the image patches extracted from the bounding boxes, to text. Text recognition is essentially a classification problem, therefore only the characters that are introduced during training can be inferred. The last decade's progress in object detection and text recognition allows us to use deep learning models for text spotting.

For the text spotting, we used the differentiable binarization text detector proposed by Liao et al. [73]. The backbone is a ResNet18 [51] neural network, which is powerful but also efficient enough to allow for fast inference.

The text recognition model is based on the work of Shi et al. [137], who proposed the CRNN architecture, that combines convolution, recurrent and transcription layers. This model can handle text of arbitrary length, is end-to-end trainable without requiring fine-tuning and is relatively small while maintaining accuracy. We use four cameras, with a coverage of 360°, to spot text.

4.2.2 Text Likelihood Maps

To incorporate text spotting into the MCL framework, we build a likelihood function of where the robot might detect a specific room number by collecting data that included image streams and the robot's pose. We apply the text spotting pipeline on the recorded images, assuming that the textual cues we are interested in follow a specific pattern ("Room X") but it can generally be used for any textual content of interest.

We compute 2D histograms for each room number, of locations where successful detections were made. The sampled locations give a sparse description of the text spotting likelihood, which we refer to as text likelihood maps (Fig. 4.2). As we are interested in a dense representation for the likelihood, we chose a simple strategy – for each text tag, we compute an axis-aligned bounding box around all sampled locations where the detection rate is above threshold τ for this textual cue. We approximate the likelihood of text detection with a uniform distribution within the bounding box.

4.2.3 Integration of Textual Cues

When a room number is detected, we store the room number and from which camera it was observed. Upon first detection, we inject particles into the corresponding area of the map (Fig. 4.2). If the last detection was made from the same camera and of the same room number, we do not inject particles. The number of particles injected is defined by the injection ratio, ρ, the number of injected particles divided by the total number of particles.

In the injection process, for a particle filter with N particles, we first remove ρN particles with the lowest weights as they represent the least likely hypotheses, and then inject an equal number of particles uniformly into the bounding box corresponding to that room number. The orientation o_i of the injected particle s_t^i depends on which camera spotted the text. We assumed that the camera detecting the text facing the room number at perpendicular angle, because a perpendicular angle usually provided the highest detection probability. Thus, we inject particles with corresponding orientation and add Gaussian noise, $\sigma_{\text{inject}} = 0.05$. The injection ratio ρ was chosen to be 0.5. A very high injection ratio could lead to localization failure if a wrong room number is detected. A low injection ratio has limited impact on the pose estimation. Therefore, the injection rate must be correlated to the confidence one has in the text prediction. The injection of particles is done asynchronously, whenever a textual cue is available, and new particles are initialized with weight $w^i = \frac{1}{N}$.

The particle injection can be seen as sampling from a proposal distribution generated by the text spotting model. This technique allows delocalized robots to get new hypotheses based on the detection, enriching their particle filters with the proposed poses, and is sometimes referred to in literature as reciprocal sampling [112].

4.3 Experimental Evaluation

The main focus of this work is an efficient, robust localization algorithm that leverage text information to better handle significant changes in the environment. We present our experiments to show the capabilities of our method and to support our claims, that we can (i) localize in quasi-static environments, (ii) localize in an environment with low dynamics, (iii) localize in different maps types – a featureless floor plan-like map, and LiDAR-based, feature rich map. These capabilities can be run online on our robot.

(a) (b) (c)

Figure 4.3. Different maps used in the experiments: (a) floor plan-like map, constructed by horizontally slicing a 3D point cloud captured with a FARO Focus X130 terrestrial laser scanner and (b) LiDAR-based occupancy grid map from GMapping [46] that was aligned to the FARO scan. (c) map built using GMapping, based on the recordings from the corridor scenario, which significantly deviates from the maps provided for localization.

Table 4.1. Algorithm parameters

σ_{odom}	σ_{obs}	r_{max}	τ	ρ	d_{xy}	d_θ
(0.02, 0.02, 0.02)	2.0	15.0 m	0	0.5	0.05 m	0.05 rad

4.3.1 Experimental Setup

To benchmark the performance of our approach, we recorded a dataset in an indoor office environment. To this end, we equipped a Kuka YouBot platform with 2 Hokuyo UTM-30LX LiDAR sensors, 4 sideways-looking Intel RealSense RGB-D (D455), and an upward-looking GoPro Hero5 Black that is used only for evaluation purposes, as shown in Fig. 3.12b. We recorded the data including wheel odometry for different scenarios.

We recorded different scenarios. A long recording was made in the corridor with all doors closed, and sequences S1-S10 are randomly sampled from that data. Similarly, the sequences starting with D are sampled from recordings D1-D4, where doors were open but contain fast-moving dynamics. We include a plot of the trajectory of the scenarios in Fig. 4.4.

To determine the ground truth pose of the robot, we use precisely localized AprilTags [104], densely placed on the ceiling of every room and corridor. The AprilTags were detected using an up-facing camera that is used solely for this purpose. The AprilTags allow to continuously and accurately localize the robot with the dedicated sensor even under dynamic changes, as explained in Sec. 3.1.

We explore two map representations, a floor plan-like map and a 2D LiDAR-based occupancy grid produced by GMapping [46], both illustrated in Fig. 4.3. The sparse, floor plan-like map was extracted from a high resolution terrestrial FARO laser scan, by slicing the dense point cloud at a fixed height. For all experiments, we use a map resolution of 0.05 m/cell and the parameters specified in Tab. 4.1.

As baseline, we compare against AMCL [111], which is a publicly available and highly-used ROS package for MCL-based localization, and our implementation of MCL that does not rely on textual cues. Additionally, we implemented two sensor models for integrating textual information into the MCL framework, referred to as SM1 and SM2. SM1 assigns all particles within the bounding box a high weight, $w_t^k = 1.0$, and a low weight, $w_t^k = 0.1$, to particles elsewhere. SM2 converts the bounding box into a likelihood map and the weight for each particle is proportional to a Gaussian applied on its distance from the bounding box, similar to

Table 4.2. Evaluation of the performance for each sequence for the corridor scenario, using the **sparse map** with 300 particles. We report ATE after convergence as angular error in radians / translational error in meters, and the time to convergence in seconds. In parentheses, the length of the sequences in seconds.

Method	S1 (234.4)	S2 (229.6)	S3 (220.8)	S4 (212.1)	S5 (203.2)	S6 (187.7)	S7 (176.3)	S8 (152.1)	S9 (145.4)	S10 (123.0)
AMCL	-/-	-/-	-/-	-/-	**0.01/0.110**	-/-	-/-	-/-	2.426/10.468	-/-
MCL	2.241/9.592	-/-	0.287/0.594	**0.045**/0.581	-/-	1.795/7.803	0.625/2.036	2.465/11.550	1.859/9.262	1.011/1.640
SM1	0.405/2.329	-/-	**0.01**/0.537	**0.045**/0.563	-/-	1.795/7.803	0.625/2.022	2.465/11.550	1.855/9.393	1.011/1.501
SM2	1.116/1.795	0.777/2.597	1.227/3.214	0.706/2.611	0.861/1.704	1.600/3.782	0.118/5.644	0.321/0.563	1.388/2.276	1.148/1.466
MCL+Text	**0.063/0.250**	**0.01/0.245**	0.063/**0.246**	0.063/**0.266**	**0.01**/0.246	**0.179/0.369**	**0.045/0.221**	**0.077/0.343**	**0.493/0.332**	**0.01/0.184**
AMCL	-	-	-	-	54.5	-	-	-	19.6	-
MCL	99.3	-	18.6	**0.0**	-	75.9	148.8	121.5	**11.5**	3.2
SM1	15.7	221.9	35.2	**0.0**	-	75.9	148.8	121.5	**11.5**	3.2
SM2	**0.0**	**1.5**	**4.7**	**0.0**	126.5	56.5	135.4	**5.6**	**11.5**	3.3
MCL+Text	**0.0**	1.7	12.6	**0.0**	**10.9**	**11.2**	**50.2**	5.7	**11.5**	**2.4**

Table 4.3. Evaluation of the performance for each sequence for the mostly static environment scenario, using the **sparse map** with 300 particles. We report ATE after convergence as angular error in radians / translational error in meters, and the time to convergence in seconds. In parentheses, the length of the sequences in seconds.

Method	D1.1 (171.4)	D1.2 (162.4)	D1.3 (144.8)	D1.4 (130.5)	D2.1 (78.2)	D3.1 (177.7)	D3.2 (160.6)	D3.3 (147.7)	D4.1 (120.0)	D4.2 (100.4)
AMCL	2.022/6.031	0.063/**0.135**	-/-	-/-	**0.010/0.095**	-/-	-/-	-/-	-/-	-/-
MCL	-/-	0.413/0.690	1.253/2.708	0.893/1.961	-/-	1.439/3.298	2.284/4.743	2.090/3.945	-/-	0.703/1.021
SM1	-/-	1.970/3.637	1.255/2.715	0.893/1.961	-/-	1.537/4.274	2.284/4.743	2.090/3.945	-/-	1.007/6.035
SM2	1.315/3.942	2.341/5.634	1.346/4.275	1.358/2.319	-/-	-/-	1.628/3.336	1.357/2.836	1.524/5.708	1.505/5.609
MCL+Text	**0.045/0.158**	**0.045**/0.175	**0.077/0.182**	**0.010/0.152**	0.045/0.279	**0.010/0.133**	**0.333/0.697**	**0.010/0.141**	**0.045/0.161**	**0.063/0.197**
AMCL	112.4	8.7	-	-	10.6	-	-	-	-	-
MCL	167.6	9.6	80.9	53.6	-	69.2	**55.2**	17.7	-	7.4
SM1	167.6	136.7	80.9	53.6	-	70.6	**55.2**	17.7	-	1.8
SM2	**2.1**	106.4	**0.2**	18.4	-	-	**55.2**	22.9	**20.4**	**0.0**
MCL+Text	2.2	**2.0**	0.7	**16.3**	**4.8**	**64.6**	55.9	**14.9**	36.4	**0.0**

(a) Sequences S1-S10

(b) Sequences D1.1-D1.5

(c) Sequences D3.1-D3.5

(d) Sequences D4.1-D4.4

Figure 4.4. Visualization of the different sequences used for evaluating our approach. Sequences S1-S10 correspond to the scenario where all doors are closed. Sequences D1-D4 were recorded with all the doors open, and with moderate amount of humans moving around. The color of the trajectory correspond to the time, where purple is the beginning and red corresponds to the end of the sequence.

Eq. (2.5). All experiments were executed with 300 particles unless mentioned otherwise, and N particles are initialized uniformly across the map.

We consider two metrics, time to convergence and absolute trajectory error (ATE) after convergence. We define convergence as the point where the prediction is within a distance of 0.5 m from the ground truth pose. If convergence did not occur within the first 95% of the sequence, then we consider it a failure, which is marked as −/−.

4.3.2 Localization under Changes using a Sparse Map

The first experiment evaluates the performance of our approach and supports the claim that we can localize in changing environment using floor plan-like maps. It is conducted on sequences recorded in a long corridor with all doors closed, while in the map these doors are all open, and it supports our claim of robust localization in face of quasi-static changes. We consider 10 sequences (S1-S10), each sequence starting at the a different location along the corridor. We evaluate the time to convergence and ATE for this challenging scenario on the 10 sequences. As can be seen in Tab. 4.2, our text-enriched method converges quickly, and outperformed the baselines in all sequences. When the map no longer reflect the environment, it is expected that classic MCL implementations would perform poorly. For text spotting sensor model to affect the pose estimation, a particle must be in the close vicinity of a specific text likelihood bounding box. With relative low number of particles, such as 300, it is unlikely to have enough particles in such a small area. Therefore, the sensor model methods have limited contribution to global localization compared to particle injection. The MCL+Text method also shows exceptional ro-

Table 4.4. Evaluation of the performance for each sequence for the corridor scenario, using the **GMapping map** with 300 particles. We report ATE after convergence as angular error in radians / translational error in meters, and the time to convergence in seconds. In parentheses, the length of the sequences in seconds.

Method	S1 (234.4)	S2 (229.6)	S3 (220.8)	S4 (212.1)	S5 (203.2)	S6 (187.7)	S7 (176.3)	S8 (152.1)	S9 (145.4)	S10 (123.0)
AMCL	-/-	-/-	**0.010/0.087**	-/-	-/-	-/-	-/-	2.436/9.151	2.420/10.756	-/-
MCL	-/-	2.399/9.822	0.941/6.763	1.352/7.113	**0.010**/0.242	**0.010**/0.547	2.176/7.869	-/-	2.600/9.385	1.280/5.901
SM1	**0.010**/0.294	0.601/3.727	0.941/6.781	1.407/7.486	2.480/11.063	**0.010**/0.525	2.176/7.869	-/-	2.600/9.385	1.918/7.056
SM2	1.347/4.181	1.344/4.644	2.014/5.334	1.618/4.257	1.002/3.598	1.566/4.100	**0.499**/1.915	0.476/1.936	2.312/8.422	1.414/4.625
MCL+Text	0.063/**0.191**	**0.063/0.203**	0.063/0.216	**0.063/0.228**	0.063/**0.171**	0.045/**0.293**	0.632/**1.788**	**0.010/0.192**	**0.697/0.920**	**0.205/0.196**
AMCL	-	-	113.0	-	-	-	-	**22.8**	19.7	-
MCL	-	29.5	21.8	**0.2**	159.1	122.8	**7.9**	-	**8.9**	**1.1**
SM1	18.1	9.7	21.8	**0.2**	173.2	122.2	**7.9**	-	**8.9**	**1.1**
SM2	6.5	1.7	**3.2**	**0.2**	114.1	10.9	**7.9**	47.8	**8.9**	**1.1**
MCL+Text	**0.0**	**1.5**	3.6	**0.2**	**8.4**	**9.7**	**7.9**	31.2	**8.9**	**1.1**

Table 4.5. Evaluation of the performance for each sequence for the mostly static scenario, using the **GMapping map** with 300 particles. We report ATE after convergence as angular error in radians / translational error in meters, and the time to convergence in seconds. In parentheses, the length of the sequences in seconds.

Method	D1.1 (171.4)	D1.2 (162.4)	D1.3 (144.8)	D1.4 (130.5)	D2.1 (78.2)	D3.1 (177.7)	D3.2 (160.6)	D3.3 (147.7)	D4.1 (120.0)	D4.2 (100.4)
AMCL	0.547/3.490	-/-	-/-	-/-	-/-	2.389/17.461	-/-	-/-	-/-	-/-
MCL	1.714/2.794	1.895/3.982	0.215/1.230	0.495/1.144	0.262/1.864	1.424/3.427	0.885/10.403	0.812/3.825	0.991/1.190	0.632/9.345
SM1	0.425/3.683	1.895/3.982	0.215/1.230	0.495/1.145	0.265/1.828	1.422/3.453	0.045/0.225	1.599/5.664	0.991/1.190	1.335/7.131
SM2	0.778/1.843	1.881/4.875	1.169/2.799	1.083/1.772	0.118/0.347	1.497/5.087	1.593/2.155	1.628/5.366	1.404/6.117	0.704/1.477
MCL+Text	**0.010/0.109**	**0.010/0.109**	**0.077/0.099**	**0.010/0.116**	**0.045/0.161**	**0.010/0.156**	**0.010/0.169**	**0.045/0.165**	**0.063/0.160**	**0.045/0.131**
AMCL	**0.0**	-	-	-	-	**35.2**	-	-	-	-
MCL	111.2	10.1	30.7	50.9	21.5	67.8	133.4	31.0	51.6	64.2
SM1	115.7	10.1	30.7	50.9	21.5	67.8	87.2	39.2	51.6	8.3
SM2	2.2	73.9	30.2	16.2	7.1	72.0	**50.3**	**15.6**	26.5	0.5
MCL+Text	2.1	**1.8**	**0.2**	**16.0**	**4.7**	71.5	50.6	19.3	**21.4**	**0.0**

bustness when reducing the number of particles in the filter, as can be seen in Fig. 4.5. The ATE for our approach is slightly larger for 10,000 particles, due to the formation of multi-modal hypotheses caused by the symmetry of the corridor.

4.3.3 Localization under Few Dynamics using a Sparse Map

The second experiment is presented to support the claim that our approach is able to localize in a floor plan-like map (not built using the robot's sensors) when the environment is mostly static. Recordings D1-D4 are taken across the lab, through different office rooms, with a small number of people moving around. In all sequences, all doors are open, and the environment is similar to the map.

This experiment considers localization in a feature-sparse map and in the presence of low dynamics. This presents its own challenges even in a mostly unchanging environment. As seen in Tab. 4.3, our text-enriched method performs best. Despite having the doors open, these scenarios include movement in a corridor with very high symmetry. Textual cues can contribute to breaking such symmetries. In addition, there are many details such as furniture, that are not part of the sparse map and can affect the accuracy of LiDAR-only localization. While SM2 shows a rather promising convergence time, the impact of the text-based sensor model is milder than particle injection, leading to divergence later on, and a large ATE.

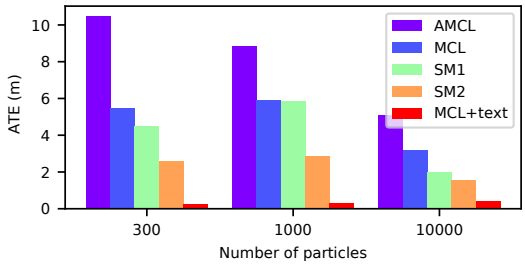


Figure 4.5. ATE (xy) averaged over sequences S1-S10 as a function of the number of particles used in the particle filter, for the different methods method. The error for MCL+Text is similar across large range of particle set sizes, exhibiting the robustness of our approach.

Table 4.6. Average inference time in ms for the sensor model on the NUC as a function of the number of particles.

	300	500	1 000	10 000
NUC10i7FNK	30	51	106	1027
Dell Precision-3640-Tower	24	40	80	793

4.3.4 Localization using LiDAR-Based Map Built with the Robot's Sensors

The third experiment is presented to support the claim that our approach is able to localize in a LiDAR-based map, when the environment is structurally changing or when there are a few dynamics in the scene. To ensure our algorithm works sufficiently well in LiDAR-based maps, we constructed a GMapping map based on 2D LiDAR scans. While this map is more detailed, the recordings were made across several weeks, resulting in some differences between the map and the environment. It is still difficult to localize globally with only 300 particles in a big scene, therefore our text-guided method enjoys an advantage. Our approach outperformed the baselines also in corridor scenario, as can be seen in Tab. 4.4. While the sensor model methods manage to converge in a timely manner (Tab. 4.4), they are less stable than our injection technique and result in greater ATE. Similarly, for the mostly static scenario, our approach achieves the best ATE overall, in addition to its fast convergence, displayed in Tab. 4.5.

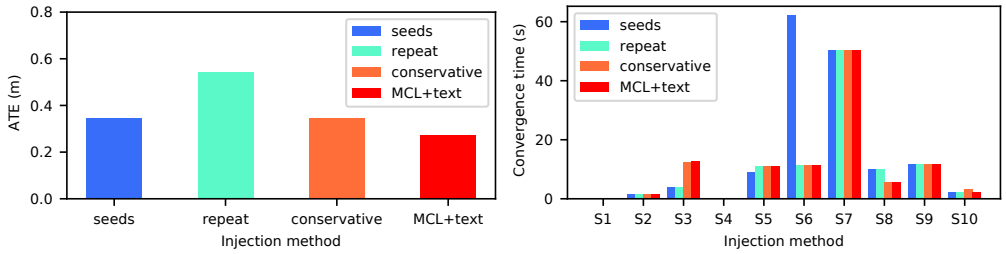
4.3.5 Runtime

The next set of experiments has been conducted to support our fourth claim that our approach runs fast enough to execute online on the robot in real-time. We, therefore, tested our approach once using a Dell Precision-3640-Tower and once on an Intel NUC10i7FNK, which we have on our YouBot. The Dell PC has 20 CPU cores at 3.70 GHz and 64 GB of RAM. The Intel NUC has 12 CPU cores at 1.10 GHz and 16 GB of RAM.

Text spotting on the NUC runs at an average of 167 ms, and on the desktop 100 ms. Tab. 4.6 summarizes the runtime results for our approach. The numbers support our fourth claim, namely that the computations can be executed fast and in an online fashion.

4.3.6 Ablation Study

Additionally, we conducted an ablation study to identify the best way of integrating the textual hints into our MCL framework. In addition to our MCL+Text method, we also explored the following strategies for injecting particles:



(a) ATE (xy) averaged over sequences S1-S10.

(b) Convergence time for sequences S1-S10.

Figure 4.6. Results for the ablation study exploring different injection strategies, with the sparse map and 300 particles.

1. Seed locations: Specific hand-picked locations in the map, which correspond to room number locations, and are used to sample particles around them with a predefined covariance.

2. Repeat: Using the text likelihood maps, described in Sec. 4.2.2, we compute a bounding box for each room number plate, and inject particles in that area for every room number detection. If we have multiple consecutive detections of a room number from the same camera, we inject particles each time.

3. Conservative: Using the text likelihood maps, we compute a bounding box for each room number plate, and inject particles in that area only once, if the filter's pose estimation mean does not lie in the bounding box. If the mean pose of MCL is within the bounding box, the filter is in line with the tag observations, and we do not inject particles. If we have multiple consecutive detections of a room number from the same camera, we inject particles only in the first detection.

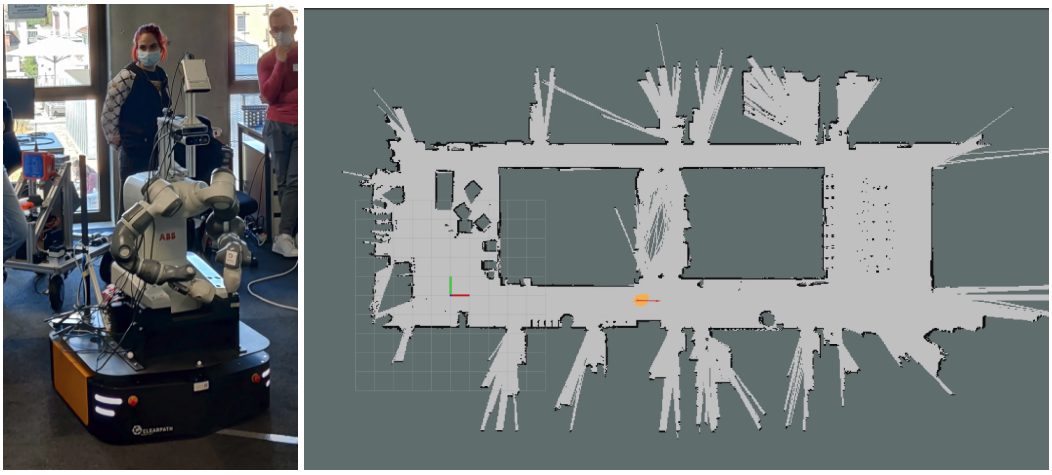
As can be seen in Fig. 4.6, MCL+Text outperforms the other text-guided methods. MCL+Text also converges faster than the other text-guided methods.

4.3.7 In-Field Experiments

In addition to the evaluation we performed in our lab, the text-based localization was also tested in different environments during Harmony project integration events. The method was tested on a Clearpath Robotics Ridgeback fitted with YuMi arms in the ASL lab in ETH (Fig. 4.7). Additionally, it was deployed on the Harmony prototype platform (Fig. 1.1) and tested in the ABB facility in Vasteras, Sweden. For both environments, we constructed the text likelihood maps manually, hand-annotating the bounding boxes without an accurate mapping procedure. Both field evaluations were considered successful by the projects reviewers, as well as our academic and industry partners.

4.4 Conclusion

In this chapter, we presented a novel approach to localize a robot in environments that deviate significantly from the provided map, as illustrated in Fig. 4.3, due to changes in the scene. Our



(a) The ETH mobile platform used for early stages of the Harmony project.

(b) Successful localization in the occupancy grid map of the ASL lab.

Figure 4.7. In-field evaluation of our proposed approach in ETH Zurich.

method exploits the readily available human-readable textual cues that assist humans in navigation. This allows us to successfully overcome localization failure in the cases where critical changes to the layout differ greatly from the map. We implemented and evaluated our approach on a dataset collected strictly for simulating such structural alterations, and provided comparisons to other existing techniques and supported all claims made in this work. The experiments suggest that incorporating human-readable localization cues in mobile robot localization systems provides considerable improvement in robustness.

Chapter 5

Exploiting Semantic Cues for Long-Term Localization

To operate autonomously in indoor environments, such as factories or offices, mobile robots must be able to determine their pose. For localization in a given map, there are two challenges: the changing nature of human-occupied environment and the quality of available maps. Precise, highly-detailed maps are an accurate representation of the environment only at the time they were captured, and they become outdated in the presence of "quasi-static" changes such as moving furniture, clutter, opening and closing doors. We describe "quasi-static" changes as long-lasting alterations (hours, days, weeks) that cause deviation between sensor observations and the given map, in contrast to dynamics such as humans and fast-moving objects. The availability of feature-rich, dense maps is not guaranteed and construction of such maps can be costly. Therefore, autonomous robots benefit from localizing in sparse maps such as floor plans or hand-crafted room layouts as they are seldom affected by changes. Architectural drawings are familiar to inexpert users and can be easily updated with CAD software. As they capture persistent structures, they typically do not require updates. However, using these sparse maps is challenging due to the paramount discrepancies between the robot's observations of the environment and the information depicted in the maps. Additionally, floor plans lack geometric information necessary to localize in a highly repetitive indoor environment, as can be seen in Fig. 5.1.

Additional sources of information can be used to overcome the challenges of global localization, and such cues have been frequently used by researchers to improve robot localization. For example, WiFi, an extremely prevalent utility, can aid in pose estimation by considering the signal strength [57]. Textual information, constantly used by humans to navigate, is readily available in human-occupied environments. However, very few works consider textual cues for localization [26, 117, 170].

Another avenue is exploiting semantic information. The last decade was marked by significant advances in object detection [12, 168] and semantic segmentation [52, 134], where semantic cues can be efficiently inferred from images (with some fine-tuning). The most commonly used map representation for robotics is an occupancy grid map [94]. However, human environments tend to be object-centric, and humans do not require precise metric information in order to navigate them [85, 166]. Rather, humans rely on a small number of specific landmarks, and associate places with the objects present there. For this reason, we consider localization in a

Figure 5.1. Floor plan maps include high degree of symmetry and low similarity to actual LiDAR measurements. This leads to multiple hypotheses that cannot be resolved correctly. We propose integrating semantic cues from a high level, abstract semantic map to assist with global localization. The red cross indicates the ground truth pose and the green dots are the particles. Left: 2D LiDAR MCL with multiple hypotheses. Right: Convergence to a single hypothesis when exploiting semantic cues, in an abstract semantic maps including various objects (colored rectangles).

sparse, approximate map, that does not require an elaborate map acquisition process. No work on semantic localization in sparse maps with abstract and hierarchical semantic information exists to our knowledge.

The main contributions of this work is a global localization system in floor plan maps that integrates semantic cues. We propose to leverage semantic cues to break the symmetry and distinguish between locations that appear similar or identical in the nondescript maps. Semantic information is commonly available in the form of furniture, machinery and textual cues and can be used to distinguish between spaces with similar layout. To avoid the complexity of building a 3D map from scans and to enable easy updates to semantic information, we present a 2D, high level semantic map. Thus, we present a format for abstract semantic map with an editing application and a sensor model for semantic information that complements LiDAR-based observation models. Additionally, we provide a way to incorporate hierarchical semantic information. Unlike most modern semantic-based SLAM approaches [22, 82, 127, 152, 163], our approach does not require a GPU and can run online on an onboard computer. Like semantic visual SLAM methods, we also rely on semantic information, but while SLAM approaches construct a map online, we focus on localization in a given map. In our experiments, we show that our approach is able to: (i) localize in sparse floor plan-like map with high symmetry using semantic cues, (ii) localize long-term without updating the map, (iii) localize in previously unseen environment. (iv) localize the robot online using an onboard computer. These claims are backed up by our experimental evaluation.

This work, titled "**Long-Term Localization Using Semantic Cues in Floor Plan Maps**" [171], was accepted to *IEEE Robotics and Automation Letters (RA-L)*, 2023.

5.1 Related Work

Recent works in extracting semantic information with deep learning models showed significant improvement in performance for both text spotting [73][137] and object detection [12][168]. In this chapter, we expand our previous work [170] to consider semantic cues via object detection, not only textual ones. The use of semantic information for localization and place recognition is applied to a variety of sensors, including 2D and 3D LiDARs, RGB and RGB-D cameras.

Rottmann et al. [124] use AdaBoost features from 2D LiDAR scans to infer semantic labels such as office, corridor and kitchen. They combine the semantic information with occupancy grid map in an MCL framework. Unlike our approach, their method requires a detailed map and manually assigning a semantic label to every grid cell.

Inspired by human navigation, Mendez et al. [84] question the importance of accurate metric data for robot localization. They use RGB-D sensors, apply semantic segmentation to RGB images and utilize the depth-images for range sensing. They globally localize in a 2D floor plan where windows, walls and doors are annotated. Building on an MCL framework, they propose an observation model that considers both range measurements and the semantic labels of detected cells. They then proceed to demonstrate that localization is still possible solely based on the semantic information, even when depth is not considered. To account for potential inaccuracies in the floor plan, their motion model includes a ghost factor to allow particles to go through "walls". Since human-oriented environments tend to contain repetitive structures, relying solely on windows, walls and doors maybe not be sufficient to enable global localization.

Hendrikx et al. [53] utilize available building information model to extract both geometric and semantic information, and localize by matching 2D LiDAR-based features corresponding to walls, corners and columns. While the automatic extraction of semantic and geometric maps from a Building Information Model (BIM) is promising, the approach is not suitable for global localization as it cannot overcome the challenges of a repetitively-structured environment.

Atanasov et al. [2] treat semantic objects as landmarks that include their 3D pose, semantic label and possible shape priors. They detect objects using a deformable part model [36], and use their semantic observation model in an MCL framework. The results they report do not outperform LiDAR-based localization.

An alternative representation for semantic information is a constellation model, as suggested by Ranganathan and Dellaert [119]. In their approach, they use stereo cameras, exploiting depth information. They rely on hand-crafted features including SIFT [77] to detect objects. Places are associated with constellations of objects, where every object has shape and appearance distribution and a relative transformation to the base location. Unlike these two approaches, our approach does not require exact poses for the semantic objects.

A more flexible representation is proposed by Yi et al. [166], who use topological-semantic graphs to represent the environment. They extract topological nodes from an occupancy grid map, and characterize each node by the semantic objects in its vicinity. It suffers when objects are far from the camera and can easily diverge when objects cannot be detected, while our approach is more robust as it relied additionally on LiDAR observations and textual cues. Similarly to the above mentioned approaches, we also use sparse representation for semantic objects. However, by using deep learning to detect objects, we are able to detect a larger variety of objects with greater confidence, and localize in previously unseen places.

SÃ¼nderhauf et al. [144] construct semantic maps from camera by assigning a place category to each occupancy grid cell. They use the Places205 ConvNet [169] to recognize places, and rely on a LiDAR-based SLAM for building the occupancy grid map. The limitation of their

approach is in the high level of semantic abstraction. As their work relied on coarse room categorization, it might not be sufficient for global localization in highly repetitive environments.

Blum et al. [11] consider the interesting case on localization in partially unstructured and drastically changing environments such as construction sites, where 2D localization is insufficient since the floors are tilted or elevated, and the flat and rectangular world assumption does not hold. The authors use foreground segmentation to extract building surfaces from images, assigning each pixel a density value. They proceed to project a 3D LiDAR scan on the image, corresponding density values to the 3D points. They follow by a weighted point-to-plane ICP, where the weight is defined by the density value, and the target pointcloud is a subset of 3D scan of the construction site. Their approach only enables tracking, and not global localization.

5.2 Approach

Our goal is to globally localize in an indoor environment represented by a nondescript floor plan and a high level semantic map. As sensors for localization, we use 2D LiDAR, cameras and wheel odometry. We build our localization approach on the Monte Carlo localization (MCL) framework [28]. To distinguish between locations that appear similar or identical in the sparse maps, we introduce imprecise, high-level semantic maps in Sec. 5.2.1 and a sensor model for semantic similarity in Sec. 5.2.2. The integration of the semantic information in the MCL framework is introduced in Sec. 5.2.3. In addition, we perform an analysis to determine the stability of semantic classes as discussed in Sec. 5.2.4 and utilize the semantic information to discard LiDAR measurements resulting from dynamic objects. Furthermore, in Sec. 5.2.5 we explore a hierarchical semantic approach for inferring the room type based on objection detection, and exploit this information to initialize the particle filter. An overview of the approach is illustrated in Fig. 5.2.

5.2.1 High-Level Semantic Maps

We represent our prior information about semantics with a 2D high level semantic map, where semantic objects are represented by a semantic class label l and a rectangle overlying the occupancy grid map. see Fig. 5.3. The size of the rectangle does not have to be very accurate and the location where it is marked can be a rough estimate of its actual placement. In our abstract map, objects differed from their actual size by 62.5%, or up to 1.25 m. This imprecise representation of semantic information is both generic enough to address variety of objects and simple enough to allow editing by end users. Each room can be assigned a name, corresponding to a text sign, and a room category representing a higher level of semantic understanding compared to basic object detection. The semantic maps can be easily created and edited using the GUI application MAPhisto[1].

5.2.2 Semantic Visibility Model

The last decade's progress in semantic interpretation allows us to use deep learning models for text spotting [73][137] and object detection [12][168]. Object detection is the task of detecting instances of semantic objects in images and videos. In our approach, the required output from an object detection model is a semantic label, a bounding box and a confidence score for every

[1] https://github.com/FullMetalNicky/Maphisto

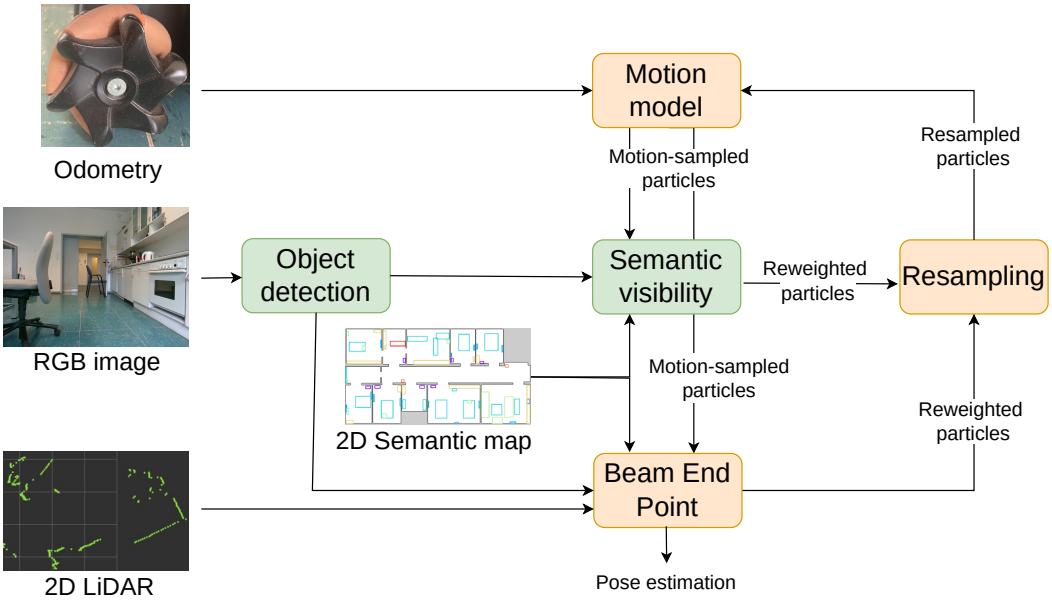


Figure 5.2. A simplified overview of the online localization approach. Given RGB images, 2D LiDAR scans an odometry input, we integrate semantic cues into an MCL framework.

detected object. For each bounding box in the prediction, we transform it to a 3D cone $\hat{\mathbf{x}}$ in the robot coordinate system, see Fig. 5.4. We take the pixel coordinates of the right and left boundaries of a bounding box, (bb_r, bb_l) and project them to 3D rays by using the camera's intrinsics and extrinsics matrices. For a pixel $\mathbf{v} = (x, y, 1)^T$, we define the associated 3D ray $\mathbf{V}(\lambda)$ as follows:

$$\mathbf{V}(\lambda) = \mathbf{O} + \lambda \mathbf{R}^{-1}\mathbf{K}^{-1}\mathbf{v}, \tag{5.1}$$

where $\mathbf{K} \in \mathbb{R}^{3\times3}$ is the camera intrinsics, $\mathbf{R} \in \mathbb{R}^{3\times3}$ is the camera rotation and $\mathbf{O} \in \mathbb{R}^3$ is camera center.

From the high-level semantic information, we construct visibility maps for the semantic classes. For each valid, free space cell c in the occupancy grid map, we compute the visibility of semantic objects. A semantic object o is visible from a grid cell c if we can ray-trace it without crossing a non-valid, i.e., occupied or unknown, cell. For each cell c, we maintain a list of all visible semantic classes. For each semantic class l, we store the set of bearing vectors,

$$\mathcal{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}, \|\mathbf{b}_i\| = 1, \tag{5.2}$$

in which objects of class l are visible. This process of constructing the visibility maps is performed once, when the algorithm is launched, and is illustrated in Fig. 5.3.

A semantic observation y_t includes the set of detected objects. For every object we store its semantic label l, its confidence score f and the center of its cone as the bearing $\hat{\mathbf{b}}$. For each particle s_t^i with pose $\mathbf{x}_t^i = (x, y, \theta)^\top$, we transform the bearing $\hat{\mathbf{b}}$ into the world coordinate system. We query the pre-built semantic visibility maps for cell c corresponding to the pose of particle s_t^i, and compare it with the observation. If an object is observed with confidence score f which is lower than a threshold τ, we ignore the observation. Otherwise, if an object with

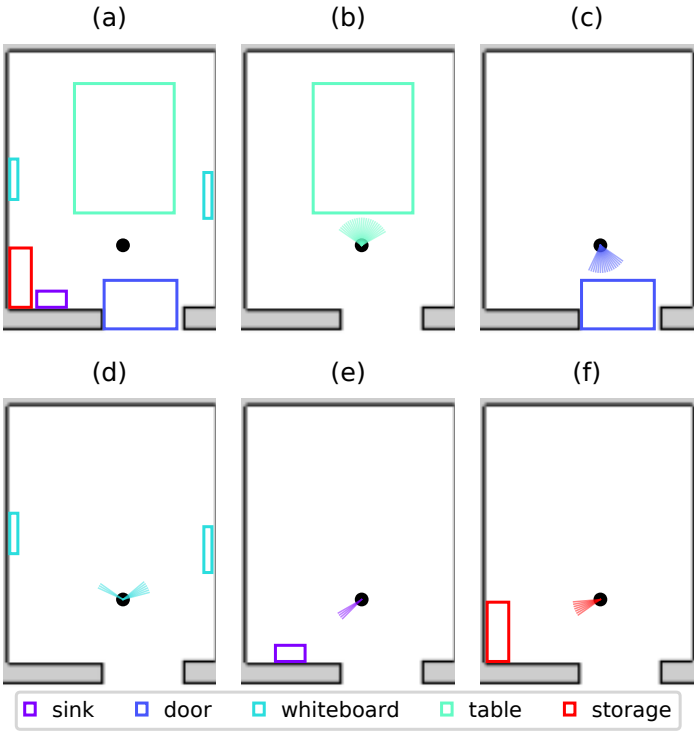


Figure 5.3. A visualization of the semantic visibility concept. (a) A semantic map of a single room, with a query point (black dot). (b)-(f) The bearings in which each semantic class objects are visible from the query point.

semantic label l is visible from cell c, we compare the observed bearing $\hat{\mathbf{b}}$ to the set of possible bearings $\mathcal{B}$ by using the cosine similarity:

$$\text{sim}(\mathbf{v}_1, \mathbf{v}_2) = \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{\|\mathbf{v}_1\|\|\mathbf{v}_2\|}, \tag{5.3}$$

where $\mathbf{v}_1, \mathbf{v}_2 \in \mathbb{R}^2$. To compare the observed bearing $\hat{\mathbf{b}}$ with all possible visible bearings $\mathbf{b}_i \in \mathcal{B}$ and select the best match according to the distance d, defined as

$$d = 1 - \max_{\mathbf{b}_i \in \mathcal{B}} (\text{sim}(\mathbf{b}_i, \hat{\mathbf{b}})). \tag{5.4}$$

For a set of detected objects z_t^S, our observation model is given by

$$p_S(z_t^k \mid \mathbf{x}_t, m_S) = \exp(-d) \qquad z_t^k \in z_t^S, \tag{5.5}$$

where z_t^k is the k^{th} confidently observed object in the set $\mathbf{z}_t^S$, and m_S is the abstract semantic map.

5.2.3 Integrating Different Modalities in the MCL Framework

We handle all information sources asynchronously – the motion model is sampled when odometry input is available, and the particles are re-weighted when an observation arrives. We integrate the 2D LiDAR measurements and the object detections using two different observation models. For a 2D LiDAR observation, z_t^L, we use the beam-end model $p_L(\mathbf{z}_t^L \mid \mathbf{x}_t, m_L)$ described

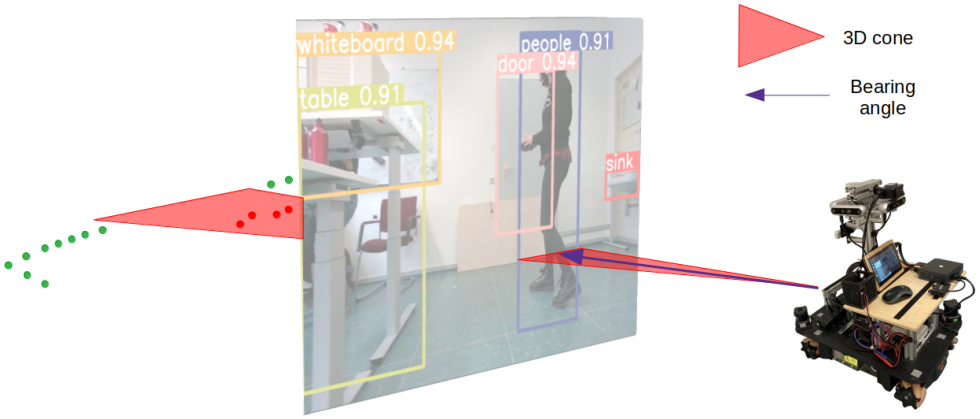


Figure 5.4. The bounding box detecting a dynamic class (person) is projected to 3D and used to mask the LiDAR beams that fall within the cone.

by Eq. (2.5). When object detection information arrives, $\mathbf{z}_t^S$, we use the semantic visibility model $p_S(\mathbf{z}_t^S \mid \mathbf{x}_t, m_S)$, detailed in Sec. 5.2.4.

The product of likelihood model assumes elements of each observation, e.g scan points in a LiDAR scan, are independent of each other. With the high angular resolution of our LiDAR this assumption does not hold. Similarly, for the semantic visibility model, detected objects are not entirely independent of each other as they often belong to the same context. The traditional product of likelihood model tends to be overconfident in such circumstances, leading us to choose the product of experts model [90], which uses geometric mean to compute the weight of each particle

$$p(\mathbf{z}_t \mid \mathbf{x}_t, m) = \prod_{k=0}^{K} p(z_t^k \mid \mathbf{x}_t, m)^{\frac{1}{K}}, \tag{5.6}$$

where z_t^k is a single component of an observation $\mathbf{z}_t$, be it a LiDAR scan or a set of detected objects. The beam-end model is triggered only when the robot moves more than d_{xy} or rotates more than d_θ, while the semantic observation model is always updated. Based on the semantic stability analysis, we detected semantic classes that tend to move frequently, which we refer to as dynamics. In addition to excluding these classes from the semantic map, we also use these detections to filter out LiDAR measurements that are the result of dynamics, as seen in Fig. 5.4

5.2.4 Semantic Stability Analysis

To decide which semantic classes would benefit localization, we estimated how likely they are to move around. We prepared a semantic map for all detectable classes, and examined the training-dedicated recordings T1-T5 spanning over multiple weeks. As our dataset includes the ground truth pose of the robot using an external reference system, we were able to conclude whether the position of detected objects corresponded to their position in the map. Using Eq. (5.4), we consider a detected object to correspond to the semantic map if $d < \tau_s$. We calculate the ratio of map-consistent detection per semantic class, and deem a semantic class stable if the ratio was above 0.6. The ratio is computed by dividing the number of map-consistent

Table 5.1. Semantic stability scores for different detected object classes computed on sequences T1-T5.

Class	sink	door	oven	whiteboard	table	cardboard	plant	drawers	sofa	storage	chair	extinguisher	person	desk
Score	0.97	0.96	0.90	0.91	0.95	0.46	0.88	0.86	0.99	0.96	0.58	0.84	0.11	1.00

Table 5.2. Algorithm parameters

Method	σ_{odom}	σ_{obs}	r_{max}	τ_s	ρ	d_{xy}	d_θ
MCL	(0.15, 0.15, 0.15)	6.0	15.0 m	-	-	0.1 m	0.03 rad
TMCL	(0.15, 0.15, 0.15)	6.0	15.0 m	-	0.5	0.1 m	0.03 rad
HSMCL	(0.15, 0.15, 0.15)	6.0	15.0 m	0.6	-	0.1 m	0.03 rad

detected objects of class l, by the total number of detections. Unstable classes are excluded from the semantic visibility model, and then stability scores are given in Tab. 5.1.

5.2.5 Hierarchical Semantic Localization

In big indoor environments, a very large number of particles is required to sufficiently cover the area in the initialization phase of global localization, which result in great computational costs. It is possible to reduce the number of used particles and achieve global localization by considering a hierarchy of semantic information. We propose to infer the room category (office, corridor, kitchen, reception) based on the predictions from the object detection. We use a nearest-neighbor classifier [95] to learn a relationship between the detected objects and the room category. We encode the semantic information as a feature vector $r \in \mathbb{R}^M$, where M is the number of classes we are able to detect. Each vector element r_l represents the number of detected objects from a specific semantic label l. We used our initial semantic observations to infer the room category, and initialize the particle filter accordingly, so that particles are only initialized in rooms of the same category. The information about the category of each room is stored in the high-level semantic map (Sec. 5.2.1).

5.3 Experimental Evaluation

The focus of this work is to provide an efficient, robust localization approach that exploits semantic information for long-term operation in sparse floor plans. We conducted our experiments to support our claims and show that our approach is able to: (i) localize in sparse floor plan-like map with high symmetry using semantic cues, (ii) localize long-term without updating the map, (iii) localize in previously unseen environment, (iv) localize the robot online using an onboard computer.

5.3.1 Experimental Setup

To evaluate the performance of our approach, we recorded a dataset on the first and second floors of our building. Our mobile sensing platform consisted of a Kuka YouBot platform with 2 Hokuyo UTM-30LX LiDAR sensors, wheel encoders, 4 cameras covering jointly a 360° field-of-view, and an upward-looking camera that is used only for evaluation purposes, seeFig. 3.12a.

The recordings span across several weeks, capturing different scenarios including moving furniture, opening and closing of doors and humans passing by.

By using precisely localized AprilTags [104], which are densely placed (approx. 1 tag/m^2) on the ceiling of every room and corridor on the second floor, we are able to extract the ground truth pose of the robot from the upwards-looking camera. The camera is used to detect the AprilTags, which allows us to accurately localize the robot even when the environment undergoes changes. The upward-looking camera captured frames at 25 Hz, and due to its wide-angle lens, we were able to detect multiple AprilTags in every frame. The pose was extracted in a least-squares fashion using multiple detections. The locations of the AprilTags were obtained using a high resolution terrestrial FARO laser scanner, and were aligned to the floor plan of the second floor. As the AprilTags were part of a large pointcloud covering the entire lab, it was possible to align the floor plan with the pointcloud. By enforcing a shared coordinate system, we are able to compare the pose estimation to our ground truth poses.

Recording R1-R11 are captured in the second floor of our building and include ground truth poses. Recording Q1-Q3 were recorded in the first floor of the building and do not include ground truth information, and are used for qualitatively evaluation on previously unseen environment. Each sequence was evaluated multiple times to account for the inherent stochasticity of the MCL framework.

In our implementation, we used YOLOv5 [58], which is a family of object detection models of varying size and performance. YOLOv5 models are capable of real-time inference on CPU-only platforms, thus making them well-suited for mobile robots. We trained a small model, YOLOv5s, on 581 images from the second floor of our building using the default training script provided in the YOLOv5 repository. The map used for localization is joint map of two CAD floor plan drawings, of the first and second floor side-by-side, illustrated in Fig. 5.1. The semantic information was integrated using our GUI application MAPhisto[2] based on our recollection of location of semantic objects. For all experiments, we use a map resolution of 0.05 m by 0.05 m per cell and the algorithm parameters specified in Tab. 5.2.

As baseline, we compare against AMCL [111], which is a publicly available and highly-used ROS package for MCL-based localization, our own MCL implementation without using semantic cues and a text-enhanced MCL [170], which we refer to as TMCL. Our method, exploiting both semantic information from object detection and hierarchical semantic knowledge discussed in Sec. 5.2.5, is referred to as HSMCL. For the tracking experiments, we considered SMCL, a variation of our approach that uses only semantic cues through the semantic visibility model, without hierarchical semantic localization. All experiments were executed with 10,000 particles in the filter unless mentioned otherwise.

We consider two metrics, the success rate and absolute trajectory error (ATE) after convergence. In our definition, convergence occurs when the estimated position is within a radius of 0.3 m from the ground truth pose and the estimated orientation is within $\frac{\pi}{4}$ rad. Tracking of the pose is considered unreliable if the pose estimate diverges for more than 1% of the time. If convergence did not occur within the first 95% of the sequence, or if the pose is not reliably tracked from convergence moment until the end of the sequence, we consider it a failure.

[2] https://github.com/FullMetalNicky/Maphisto

Table 5.3. Evaluation of the performance on sequences recorded all across the second floor in the span of several weeks. We report ATE after convergence as angular error in radians / translational error in meters, and the success rate.

Method	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	AVG
AMCL	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-
MCL	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-
TMCL	-/-	-/-	0.048/0.16	-/-	-/-	-/-	0.034/0.22	0.043/0.18	0.050/0.21	-/-	0.034/0.18	0.042/0.19
HSMCL	0.054/0.15	0.064/0.24	0.069/0.25	0.205/0.23	0.100/0.34	0.064/0.23	0.069/0.23	0.049/0.18	0.090/0.26	0.052/0.16	0.052/0.25	0.079/0.23
AMCL	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
MCL	40%	20%	60%	40%	20%	0%	0%	40%	20%	60%	0%	27%
TMCL	80%	0%	100%	80%	60%	40%	100%	100%	100%	80%	100%	76%
HSMCL	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%

Table 5.4. ATE for tracking on a subset of sequences recorded all across the second floor in the span of several weeks. The particle filter was set to adaptive 1,500-5,000 particles for AMCL and a fixed 1,500 particles for MCL and SMCL. Angular error in radians / translational error in meters.

Method	R3	R4	R6	R7	R8	R10	AVG
AMCL	-/-	-/-	0.047/0.22	-/-	-/-	-/-	0.047/0.22
MCL	0.051/0.17	0.050/0.21	0.051/0.29	0.064/0.23	0.039/0.14	0.041/0.15	0.049/0.20
SMCL	0.063/0.21	0.046/0.22	0.068/0.29	0.048/0.19	0.042/0.15	0.044/0.13	0.052/0.20

5.3.2 Long-Term Localization in CAD Floor Plans

The first experiment evaluates the performance of our approach and supports the claim that we are capable of long-term localization in sparse, floor-plan-like maps. Sequences R1-R11 are recorded in April-June 2022, and traverse all the rooms in the second floor. The given map had been constructed in 2021. All sequences include humans walking around, opening and closing of doors, moving furniture and large amount of clutter. We repeat the evaluation of each sequence 5 times, computing the success rate, ATE and convergence time over all 5 runs, and compare against the baselines. As can be seen in Tab. 5.3 the semantically-enhanced methods have superior performance over the baselines. AMCL and MCL are mostly used with detailed maps constructed using range-sensor measurements, and we can attribute their poor performance to the sparse nature of the floor plans. This highlights the impact of semantic information when localizing in nondescript, sparse maps, especially in face of high geometric symmetry.

As reported in Tab. 5.3, upon successful convergence, HSMCL achieves accuracy of 0.23 m and negligible angular error. While TMCL has lower ATE, it only converges in 76% of the cases. HSMCL successfully converges, on average over all sequences, after 25 s.

We further provide pose tracking experiments. A similar approach to ours, Boniardi et al. [14], tracked the pose of a robot by inferring the room layout from camera images, reporting RMSE of approx. 0.23 m and approx. 0.04 rad with adaptive particle number ranging between 1,500-5,000. However, they did not provide open source code. Our office environment is similar to the Freiburg one where Boniardi et al. [14] evaluated their method. For the tracking experiments we used SMCL, which integrates semantic cues from object detection, without hierarchical information. We report our tracking results with fixed 1500 particles in Tab. 5.4, achieving an ATE of 0.2 m and 0.05 rad. This suggest that integrating semantic cues, and specifically, our SMCL approach, are beneficial also for tracking purposes and not only for global localization.

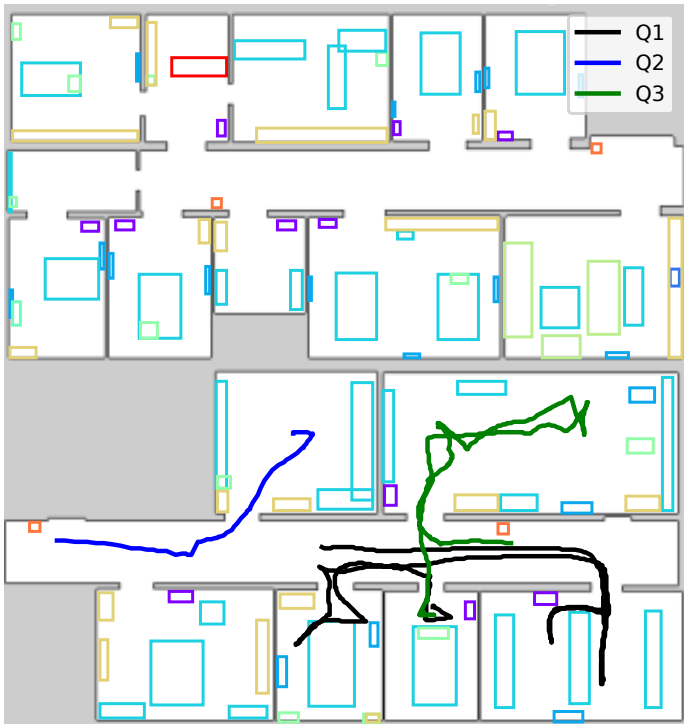


Figure 5.5. Examples of pose estimation for localization in previous unseen environment, using SMCL and 10,000 particles.

5.3.3 Localization in a Previously Unseen Environment

To support our claim that we are able to localize in a previously unseen environment, we qualitatively evaluate our method on sequences Q1-Q3 recorded on the first floor of our building. The object detection model and the room category classifier were not trained or validated on data from this floor. While the first floor is not entirely dissimilar to the second one, it does include different furniture and rooms that serve different purposes such as a classroom and a robotics lab. The pose estimated by SMCL sequences is shown in Fig. 5.5. Our approach correctly predicts that the robot is located in the first floor and identifies the correct room and maintaining a trajectory that is consistent with floor plan map. We manually verified that the robot's estimated trajectory corresponded to the rooms visited in the video recordings.

5.3.4 Ablation Study

To justify our use of both low-level and hierarchical semantic information, we conducted an ablation study. We analyzed three strategies for integrating semantic knowledge into an MCL framework. SMCL uses only semantic cues through the semantic visibility model. HMCL uses semantic hierarchy, described in Sec. 5.2.5, to initialize the particles only in the rooms corresponding to the observed room category, and then relies solely on the LiDAR information. HSMCL combines both strategies. The ATE was computed only on stable sequences with 100% success rate. As can be seen in Tab. 5.5, utilizing the two levels of semantic information benefits

Table 5.5. Performance on 11 sequences recorded all across the second floor in the span of several weeks. A run was considered successful if the algorithm converged to the ground truth in the first 95% of the recording and remained localized until the end of the sequence. Angular error in radians / translational error in meters.

Method	Hierarchy	Semantics	Success	ATE (# of stable sequences)	ATE (# of successful runs)
MCL			27%	- (0)	0.046/0.20 (15)
HMCL	✓		61%	0.046/0.21 (3)	0.044/0.19 (34)
SMCL		✓	81%	0.055/0.23 (7)	0.066/0.24 (45)
HSMCL	✓	✓	100%	0.079/0.23 (11)	0.079/0.23 (55)

Table 5.6. Runtime for HSMCL, with 10,000 particles. The Yolov5s results are for inference on a single camera.

Platform	Sem. Visibility	Beam-End	Yolov5s (640x480)	Yolov5s (320x240)
NUC10i7FNK	55 ms	24 ms	223 ms	57 ms
Dell Precision-3640-Tower	19 ms	14 ms	10 ms	6.8 ms

localization. HSMCL was able to localize stably even on the challenging sequences, where other methods failed. The ATE for HSMCL is on par with the other methods, and the slightly larger error can be attributed to including more challenging sequences and runs in the computation of the ATE for HSMCL, sequences and runs where other methods failed to localize entirely.

5.3.5 Runtime

We evaluate the runtime performance of our approach in support of our fourth claim, that we are able to operate onboard and allow real-time localization. We tested our approach on a Dell Precision-3640-Tower (with NVidia GeForce RTX 2080) and once on an Intel NUC10i7FNK, which we have on our robot. The Dell PC has 64 GB of RAM and runs at 3.70 GHz. The Intel NUC has 16 GB of RAM and runs at 1.10 GHz. The measurements are reported in Tab. 5.6. Since we are using 4 cameras simultaneously for object detection, we used an optimized ONNX export of YOLOv5s, and run inference on 320 by 240 images. Qualitative online tests indicates that reducing the resolution does not impact the detection accuracy significantly. These runtime results suggest that our approach is suitable for online localization, and utilizes semantic information without requiring a GPU onboard.

5.3.6 In-Field Experiments

In addition to the evaluation we performed in our lab, our semantic-guided localization was also tested in different environments during Harmony project integration events. The method was tested on a Clearpath Robotics Ridgeback fitted with YuMi arms in the ASL lab in ETH (Fig. 4.7). Additionally, it was deployed on the Harmony prototype platform (Fig. 1.1) and tested in the ABB facility in Vasteras, Sweden. For both environments, the semantic maps were hand-annotated using the MAPhisto GUI application. Both field evaluations were considered success-

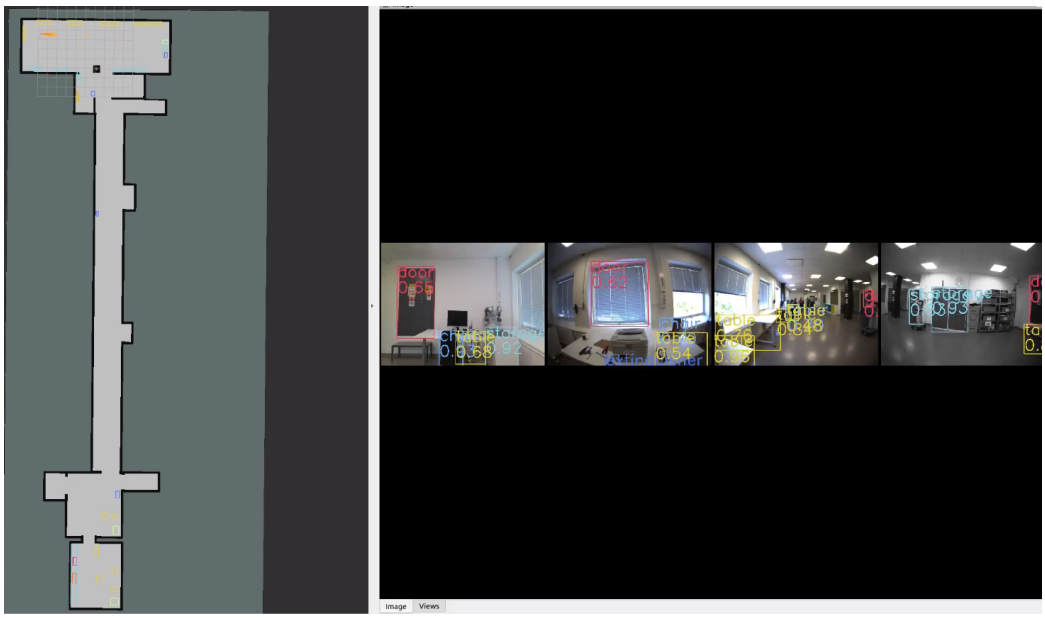


Figure 5.6. In-field evaluation of our proposed approach in the ABB facility.

ful by the projects reviewers, as well as our academic and industry partners.

5.4 Conclusion

Our approach incorporates semantic information, from low-level object detection to higher understanding of room categories, to assist navigation in human-oriented environments. This enables us to successfully localize in sparse floor plans under high geometric symmetry and changing environments. We demonstrate that using sparse and abstract map representation benefits long-term localization, and reduces the need to update the map. We also provide a tool for updating the semantic map, when critical changes occur. For our evaluation, we recorded a dataset spanning across weeks, introducing a variety of elements that are not represented in the floor plan, and the changes a human-occupied environment undergoes. We compared our performance to other existing methods, supporting all of our claims. The results of our experiments imply mobile localization systems can benefits greatly from exploiting ever-present semantic cues.

Chapter 6

Enriching Floor Plans with 3D Metric-Semantic Information

In the previous chapters, we focused on the issue of robot localization. In this chapter, we introduce and address the task of map construction, which is equally important and essential for mobile autonomous systems. Object-based maps, coupled with semantically-augmented localization are the foundation for more complex robotics tasks such as navigation and manipulation, as well as AR/VR applications. They enable to estimate the 3D geometry of the environment, and enrich it with semantic information. We focus on the metric-semantic map construction coupled with long-term object-based localization, using only monocular RGB frames and a floor plan prior. We use RGB cameras instead of RGB-D due to their lower power consumption and bandwidth requirements. For both tasks, we are interested in approaches that operate online on a mobile platform.

Previous works on 3D mapping leverage 3D reconstruction techniques to create a geometric description of the environment [103, 107]. In recent years, the progress in semantic segmentation and object detection enabled the integration of semantic information into 3D reconstruction [45, 64, 122]. Some works have put emphasis on highly-accurate and detailed reconstruction of the environment [71, 128]. Other approaches use 3D bounding boxes as a valuable and compact abstraction for objects [70, 165]. Here, we leverage 3D object detection for enriching a floor plan map with metric-semantic information suitable for long-term object-based localization. Floor plan maps are commonly available and store information about unchanging structures, such as walls. Localization in floor plan maps is challenging due to the sparsity of geometric information; its advantage, however, is that information stored in floor plans rarely change. Often, additional sources of information are used to support robust localization, such as textual cues [170] and WiFi signals [57]. 3D object-based maps can be used to improve long-term localization in floor plan maps. Works such as the ones by Li et al. [70, 71] allow the construction of object-based maps from monocular RGB frames, but they do not include static structural elements such as walls, which have a critical importance for localization and planning tasks.

The main contribution of this work is a global localization and object-based mapping system using 3D semantic information suitable for long-term operations in dynamic environments. We address the difficulty of creating 2D and 3D labels by proposing an efficient method for label generation from RGB frames. These labels can then be exploited to achieve accurate perfor-

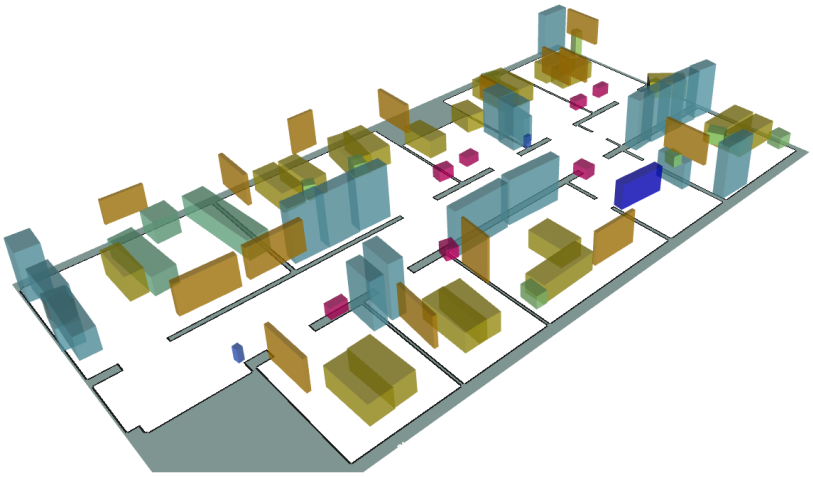


Figure 6.1. A 3D semantic metric map combining a floor plan with 3D object bounding boxes built using our approach. This map is used for long-term localization in dynamic indoor environments. Different box colors indicate different object classes.

mance on the target environment by finetuning off-the-shelf detection models. We analyze the performance of our detector, creating a probabilistic detection model that benefits both, map creation and object-based global localization. We utilize 3D object detection to construct object-centric maps, as seen in Fig. 6.1, augmenting readily-available floor plan maps with semantic information. We provide a global localization system for the pre-built object-based map with an uncertainty-aware sensor model for 3D object information, relying solely on monocular cameras.

In sum, our approach is able to (i) generate 3D labels for fine-tuning of 3D object detection models, (ii) enrich floor plans with object information, (iii) and localize in such maps in an online fashion, using onboard computers. These claims are backed up by our experimental evaluation.

This work, titled "**Constructing Metric-Semantic Maps Using Floor Plan Priors for Long-Term Indoor Localization**" [172], was accepted to *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2023.

6.1 Related Work

6.1.1 3D Object Detection

Scene understanding is the ability of recognizing objects and obtaining a semantic interpretation of the surrounding environment. In the last years, deep learning enabled tremendous advancements in image-based object detection [43, 120], semantic segmentation [134], and panoptic segmentation [23, 63, 140]. In the 3D domain, object detection approaches aimed to reproduce the efforts of 2D object detection in 3D [138, 153]. Most approaches, however, took alternative paths. Qi et al. [115] propose an end-to-end 3D object detection network inspired by the generalized Hough voting. The authors also propose an extension fusing 2D and 3D voting for boosting 3D object detection [116]. Recent efforts in 3D object detection tackle the

problem by processing more than one frame at a time [70, 155], since performances of single-view approaches [71] are degraded by the depth-scale ambiguity [70]. Recently, Brazil et al. [16] proposed a single-view approach, called Cube R-CNN, that achieves state-of-the-art results for 3D object detection and solves the depth-scale ambiguity by introducing a training objective that incorporates a virtual depth.

6.1.2 Semantic Mapping

Map construction is a crucial elements of most robotics applications, and object-based maps are a step further towards higher level scene understanding. In recent years, learning-based approaches were applied to 3D object-based map construction. Many works were aimed at object-aware mapping [45, 64] or simultaneous localization and mapping (SLAM) [82, 142] with RGB-D cameras. In this work, we focus on object-based mapping with known poses using monocular RGB images.

For this task, Li et al. [70] extend the 2D object detection model DETR [19] with another head that predicts 3D bounding boxes. They train a graph neural network for data association, which considers the 3D bounding boxes, 2D bounding boxes and semantic classes of detected objects, and matches them to map objects. Once a prediction is matched to a map object, they rely only the 2D bounding box information to optimize a multi-view 3D bounding box represented as a super-quadratic surface. While they claim learning features for data association is preferable to hand-crafted measures like IoU, it can also impose an additional refinement phase when deploying the mapping algorithm in a new environment, additionally to the fine-tuning an object detection model might require.

Li et al. [71] propose an approach for semantic-aware reconstruction. They operate on RGB imaegs extracting 2D bounding boxes and 3D object poses using a modifies DETR [19] model. Each cropped patch containing an object detection is mapped into a shape code, and detections are tracked using Bayesian filtering. Hungarian algorithm is used for the data association, based on the predicted object location and the shape code. They perform a coarse-to-fine reconstruction, first representing the objects only as localized bounding boxes, and using shape codes for a detailed reconstruction, using DeepSDF [110].

A similar approach is the one introduced by Runz et al. [128], who apply instance segmentation to RGB images and use ray clustering for data association. 3D rays from the camera center through the center of a 2D bounding boxes are expected to approximately intersect if they correspond to the same detected object. They train a shape encoder to infer object shape code from a single-view, and fuse multiple associated single-view codes to reconstruct a dense surface. These works focus on detailed reconstruction of indoor spaces, resulting in maps that are not suitable for long-term localization, as they introduce objects that are transient instead of filtering out noise that can negatively impact robust localization.

6.1.3 Semantic Localization

Long-term global localization in changing environments is a challenging task. Researchers investigated different sources of information such as textual cues [26, 170], wifi signal strength [89], and semantic cues to support pose estimation. Rottmann et al. [124] use a semantically-labeled occupancy grid map to localize in a Monte Carlo Localization (MCL) [28] framework, performing place classification from RGB frames. Mendez et al. [84] extract semantic information about structural elements (walls, windows, doors) from floor plans, classify the pixels of an RGB-D

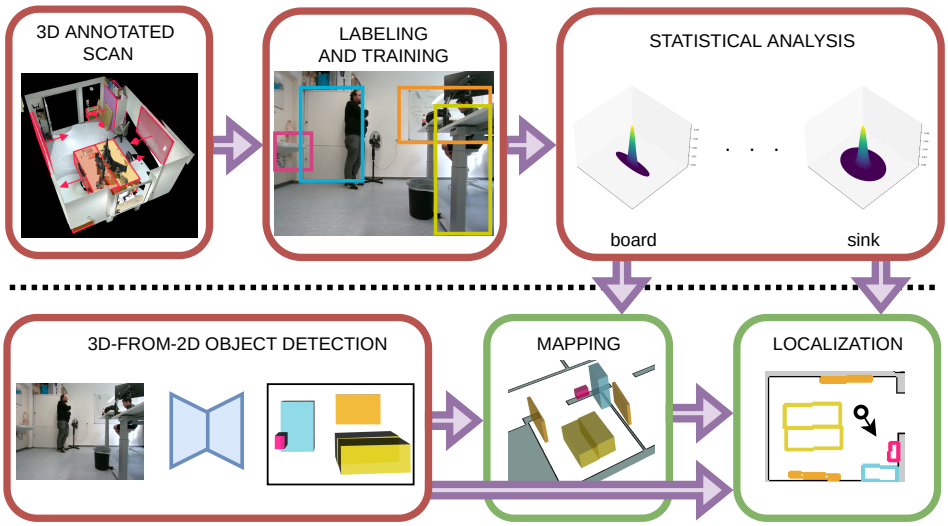
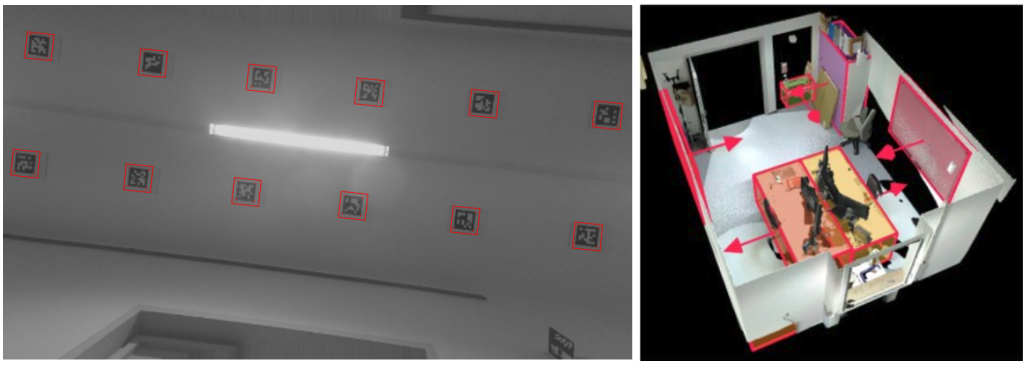


Figure 6.2. An overview of our approach. Top row: offline pre-computation to adapt the approach to a specific environment. Bottom row: 3D-from-2D object detection and mapping, that can be executed on demand when the environment undergoes structural changes and a map update is necessary.

image and match it against the semantic floor plan to localize. Both approaches do not use an explicit object-based map. Atanasov et al. [2] use objects as landmarks, defined by their 3D pose, semantic class and possible shape priors. The object are detected using a deformable part model [36], and the semantic information is integrated into a MCL framework. Ranganathan and Dellaert [119] suggest a different representation for a semantic map, by using a constellation model. They detect objects using hand-crafted features, and rely on depth information provided by stereo cameras. Yi et al. [166] localize using a topological graph, where each node is characterized by the semantic objects in its vicinity. Similarly to these approaches, we incorporate the prediction uncertainty into our localization approach and use sparse object-based maps. We provide a pipeline for acquiring the metric semantic map. Most other approaches use 2D object detection, while we utilize 3D object information. Zimmerman et al. [171] incorporate high and low-level semantic cues from 2D object detection and geometric information from 2D LiDAR scanners, but use manual map creation. In contrast to this, our approach does not require LiDAR input.

6.2 Approach

We aim to enhance sparse floor plans with semantic cues and globally localize in these object-based maps using monocular cameras. In Sec. 6.2.1, we present a way of creating 3D labels for fine-tuning 3D object detection models. Using such labels, we show how to fine-tune an object detection model and learn a noise model, see Sec. 6.2.2, and use it to build a probabilistic object-based map. We construct a standard metric-semantic map using 3D object detections on posed RGB images, detailed in Sec. 6.2.3. We exploit the object-based maps for object-based global localization, as described in Sec. 6.2.4. An overview of our approach is visualized in



(a) (b)

Figure 6.3. Our global localization infrastructure can enable automatic generation of 3D labels by computing the relative pose between the robot and the observed objects. (a) Extracting the robot's pose using the AprilTag infrastructure. (b) One-time labeling of the objects in a 3D scan of the lab.

Fig. 6.2.

6.2.1 Label Generation for 3D Object Detection

The performance of off-the-shelf 3D object detection models are often not accurate enough for the purpose of map building and they possibly focus on classes that are not beneficial for the purpose of indoor localization. For this reason, mapping and localization systems might need to finetune an existing model to the environment of interest. 3D object detection from RGB images requires 3D bounding boxes annotations that include the object dimension, rotation and translation relative to the camera frame, in addition to 2D bounding box annotation for the RGB image with semantic category. Often, truncation and visibility of objects are also needed. Truncation refers to the percentage of the object in the camera frustum, while visibility is a measure of occlusion of an object by other object in the scene. Providing accurate labels in a real-world environment is challenging. In our approach, we construct the 3D labels based on a 3D model of the environment extracted from a 3D scan, 2D object detector and posed RGB images. As a pre-processing step, we manually annotate the 3D model of the environment with 3D bounding boxes of objects of interest. Relying on our accurate localization infrastructure, we can extract the relative poses and the 3D bounding boxes of observed objects (Fig. 6.3). We project the 3D bounding boxes onto the posed camera frame using

$$\mathbf{x} = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \mathbf{X}, \tag{6.1}$$

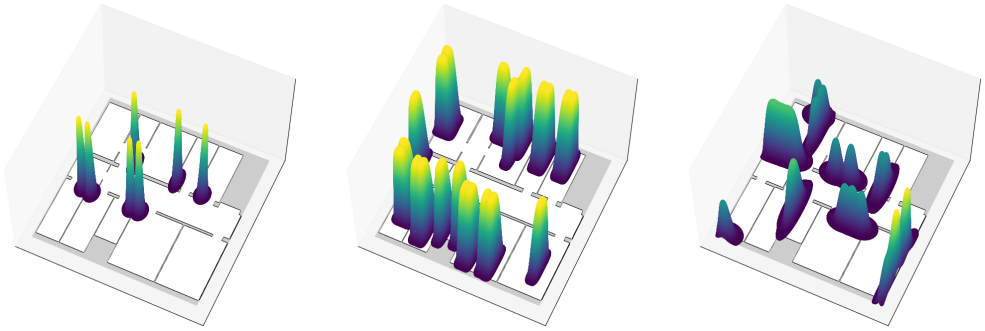
where $\mathbf{X} = (x, y, z, 1)^\top$ is a 3D point in the world coordinate system in homogenous coordinates, $\mathbf{K} \in \mathbb{R}^{3\times3}$ is the camera calibration matrix, $\mathbf{R}$ and $\mathbf{t}$ are the camera rotation matrix and translation vector, and $\mathbf{x}$ is a point in the image plane in homogenous coordinates. We then determine the visibility and truncation for every annotated object. To determine if an object is occluded by a dynamic obstacle, such as a person or a closed door, or by a static obstacle that is not annotated, we use a 2D object detector, as seen in Fig. 6.4. The detector is trained on classes of interest including dynamics and is made available by Zimmerman et al. [171]. For every object detected

Figure 6.4. 2D object detection to create better 3D annotations. Top: when rendering the ground truth objects from the camera, we have no information about dynamic objects like closed doors, which results in wrong annotations. The 2D object detection detects a door, and none of the objects in the 3D map. Therefore, no 3D annotations are generated (faded colors). Bottom: the 2D object detection detects tables and boards, but not the drawers due to occlusion. Therefore 3D annotations are generated only for the boards and the tables.

using the 2D object detector, we match a previously-annotated 3D bounding box based on the semantic class and the IoU between the projected and the detected 2D bounding box. In this way, for every posed RGB frame, the system annotates the detectable 3D objects, including their relative pose in the camera frame, dimensions, semantic class, 2D bounding boxes, visibility and truncation. Using these labels, we fine-tune a 3D object detector. We choose Cube R-CNN [16], but our implementation allows integration of any other 3D object detector with the same output structure. Further details about the architecture and training procedure can be found in Sec. 6.3.

6.2.2 Statistical Analysis of 3D Object Detections

Given posed RGB frames, capturing objects belonging to classes of interest, we run inference on the images using Cube R-CNN, fine-tuned with our labels. Given two non-overlapping object predictions of the same class, a key issue is determining whether they belong to the same instance suffering from noisy detection or they are two separate instances in close vicinity. This is particularly common with semantic classes whose bounding boxes have one dimension which is substantially smaller than the other, such as whiteboards. To address this problem, we analyze the per-class characteristic noise for detections. For each class, we build a 2D probability distribution by matching the 3D predictions to map objects $\mathbb{O}$, and projecting them in 2D on the ground plane. For matching, a predicted object can be assigned to a ground truth object only if the euclidean distance between the center of the predicted object $\mathbf{c}_o$ and the center of the ground truth object $\mathbf{c}_{GT}$ is smaller than a threshold δ.



(a) Sinks (b) Tables (c) Cabinets

Figure 6.5. Object probability map m_p which contains the per-object distribution $p_o(c \mid l)$, for specific classes of interest.

In the case a predicted object can be assigned to multiple ground truth objects, we select the one with the highest 3D IoU. In case the prediction has no overlap with any ground truth, we match based on center distance only. Additionally, if no ground truth center is within distance δ from the center of the predicted object, we discard the prediction. Then, we aggregate predictions of different objects of the same class by transforming them into an object-centric coordinate system where the center of the associated ground truth object is the origin. For every class, we take the matched predictions and project their center on the 2D plane. Additionally, we discretize the 2D plane into cells of 0.05 m creating a histogram of occurrences. Then, we fit a Gaussian distribution

$$p(\mathbf{c} \mid l) = \frac{1}{\sqrt{2\pi\,|\boldsymbol{\Sigma}|}} \exp\Big\{-\frac{(\mathbf{c}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{c}-\boldsymbol{\mu})}{2}\Big\}, \tag{6.2}$$

where $\mathbf{c}$ indicates the 2D center coordinate of the predicted object, and l is its semantic class.

Then, we transform the per-class distribution $p(\mathbf{c} \mid l)$ to be centered around each map object o of class l. We shift the mean $\boldsymbol{\mu}$ and rotate the covariance $\boldsymbol{\Sigma}$ of the class-specific distribution in local frame, according to the projected 2D center point $\mathbf{c}_o$ and the rotation matrix $\mathbf{R}_o$ of an object:

$$\boldsymbol{\mu}_o = \boldsymbol{\mu} + \mathbf{c}_o, \qquad \boldsymbol{\Sigma}_o = \mathbf{R}_o\,\boldsymbol{\Sigma}\,\mathbf{R}_o^\top. \tag{6.3}$$

Thus, the per-object distribution $p_o(\mathbf{c} \mid l)$ is given by the parameters $\boldsymbol{\mu}_o$ and $\boldsymbol{\Sigma}_o$ of Eq. (6.3).

We build the object probability map m_p by composing the individual Gaussians of the map objects $p_o(\mathbf{c} \mid l), \forall o \in \mathbb{O}$. Our object probability map m_p can be seen in Fig. 6.5.

6.2.3 3D Semantic Map Construction

Given posed RGB images and a 3D object detector, we construct a metric semantic map to enhance a floor plan map. We define an object as

$$o = \{\mathbf{c},\, \mathbf{D},\, \mathbf{R},\, \mathrm{I}_{\text{active}},\, n_{\text{skip}},\, n_{\text{match}}\}, \tag{6.4}$$

where $\mathbf{c}$ is the center of the object, $\mathbf{D} = (W, H, L)$ is the object's bounding box dimensions, $\mathbf{R}$ is the orientation of the bounding box, $\mathrm{I}_{\text{active}} \in \{0, 1\}$ is a state that indicates whether an object

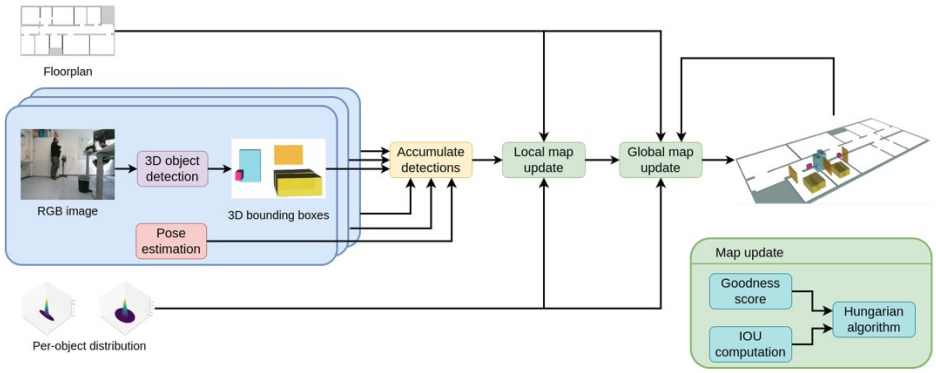


Figure 6.6. Flow diagram for the 3D metric-semantic map construction.

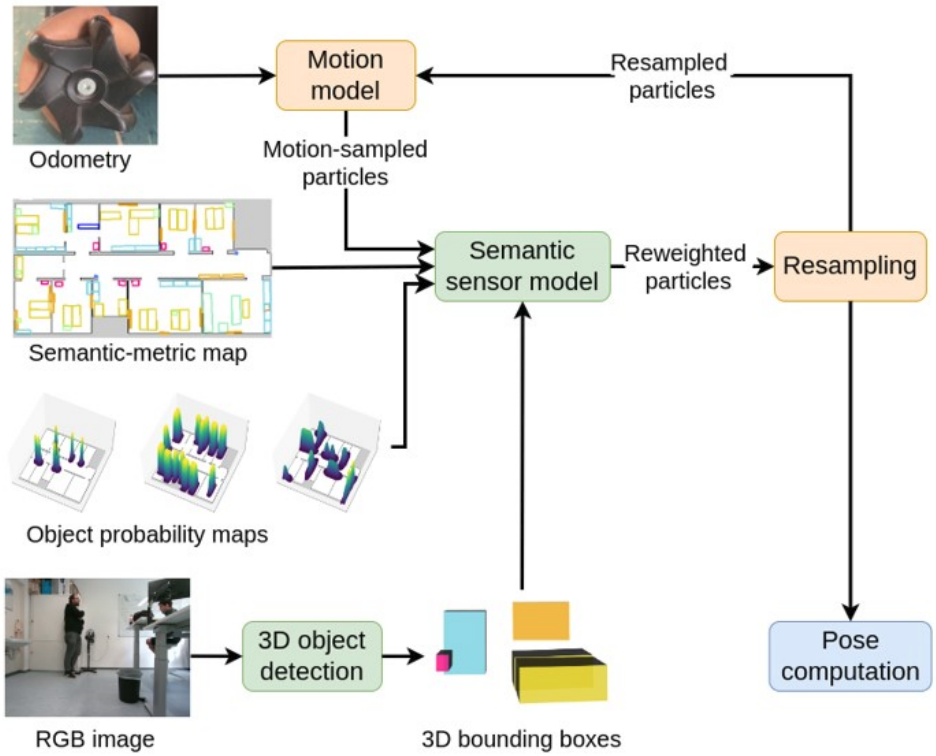


Figure 6.7. Flow diagram for the 3D semantic localization approach.

is active or not, n_{skip} and n_{match} are two object-specific counters that will be explained in the following. An object is in the active state if it is in the camera frustum and is not occluded by other objects. We test for the visibility of an object by rendering the 3D scene into the camera frame, constraining the visibility with walls extracted from the floor plan map. We aggregate consecutive detections into a short-term, local map $\hat{m}_g$ by associating detected objects across different frames. Active objects are associated by means of the Hungarian algorithm [67] and a cost function defined by

$$\mathcal{C}(o_1, o_2) = \frac{1}{2}\left(\mathcal{C}_{\text{IoU}} + \mathcal{C}_{\text{cen}}\right) \tag{6.5}$$

$$\mathcal{C}_{\text{IoU}} = 1 - \text{IoU}(o_1, o_2) \tag{6.6}$$

$$\mathcal{C}_{\text{cen}} = 1 - p_{o_1}(\mathbf{c}_{o_2} \mid l_{o_2}), \tag{6.7}$$

where o_1 and o_2 are detected objects, $p_{o_1}(\mathbf{c}_{o_2} \mid l_{o_2})$ represents the goodness score of a center prediction based on the statistical analysis in Sec. 6.2.2, $\mathbf{c}_{o_2}$ and l_{o_2} are the center and the semantic class of object o_2, respectively. After the robot has moved more than d_{xy} or rotated more than d_θ, we integrate the objects of $\hat{m}_g$ into the global object map m_g using the matching strategy described above. Given the associations computed by the Hungarian algorithm, we merge matched objects if the cost in Eq. (6.7) is below a threshold τ_{cost}. Otherwise, we initialize a new map object. When objects are merged, we increase the n_{match} count. If an active map object was not associated with a prediction, we increase the n_{skip}. When merging a prediction o_1 to a map object o_2, we use a weighted average to update the center and the dimensions of the bounding box:

$$\begin{aligned} \mathbf{c}_{o_2} &= \frac{n^{o_2}_{\text{match}}\,\mathbf{c}_{o_2} + n^{o_1}_{\text{match}}\,\mathbf{c}_{o_1}}{n^{o_1}_{\text{match}} + n^{o_2}_{\text{match}}} \\ \mathbf{D}_{o_2} &= \frac{n^{o_2}_{\text{match}}\,\mathbf{D}_{o_2} + n^{o_1}_{\text{match}}\,\mathbf{D}_{o_1}}{n^{o_1}_{\text{match}} + n^{o_2}_{\text{match}}}. \end{aligned} \tag{6.8}$$

We update the rotation matrix by computing the weighted average and extracting the rotation matrix via SVD as proposed by Moakher [91]:

$$\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top = \text{SVD}\left(\frac{n^{o_2}_{\text{match}}\,\mathbf{R}_{o_2} + n^{o_1}_{\text{match}}\,\mathbf{R}_{o_1}}{n^{o_1}_{\text{match}} + n^{o_2}_{\text{match}}}\right) \tag{6.9}$$

$$\mathbf{R}_{o_2} = \mathbf{U}\mathbf{V}^\top. \tag{6.10}$$

Additionally, we update the weight of the object by summing up n_{match}. We purge objects from the map when

$$\frac{n_{\text{match}}}{n_{\text{skip}}} < \tau_{\text{purge}},$$

where τ_{purge} is an empirically-chosen threshold.

We obtain room segmentation by applying morphological operations and connected-component analysis on the floor plan. This allows us to associate objects to rooms, so we update only objects located in the room we are currently in. A summary of the mapping approach is brought in Fig. 6.6.

Table 6.1. Algorithm parameters

Method	σ_{odom}	σ_{obs}	$r_{\max}$	τ_s	d_{xy}	d_θ
Ours	(0.15, 0.15, 0.15)	-	-	-	0.1 m	0.03 rad
HSMCL	(0.15, 0.15, 0.15)	6.0	15.0 m	0.6	0.1 m	0.03 rad

6.2.4 3D Semantic Localization

We globally localize using an MCL [28] framework. Our sensor model is based on the probabilistic analysis of the accuracy of the 3D object detection model, see Sec. 6.2.2. Our observation $\mathbf{z}$ includes the object class l, the confidence score f and the object 3D bounding box $\mathbf{b}_{\text{3D}}$ in the coordinate frame of the camera. For every particle s_t, we transform the prediction into world frame using the particle state $\mathbf{x}_t$, and we project it on the ground, obtaining the corresponding 2D bounding box $\hat{\mathbf{b}}_{\text{2D}}$.

To compute the weight of every particle we consider two measures – the object-based likelihood $p_o(\mathbf{z} \mid m_p, \mathbf{x}_t)$ and the shape similarity score p_s. Using the object probability map m_p computed in Sec. 6.2.2, we sample p_o at the location indicated by the center $\mathbf{c}$ of the transformed 2D bounding box. To compute the likelihood of predicting a center $\mathbf{c}$ of class l, we consider the corresponding class distribution for every object given by the object probability map m_p:

$$p_o(\mathbf{z} \mid m_p, \mathbf{x}_t) = \max_{\{o \in \mathbb{O} \mid l_o = l\}} p_o(\mathbf{c} \mid l), \tag{6.11}$$

where l_o is the semantic class of object o. For data association, we match the prediction to map object with the highest likelihood, referred to as $o_{\max}$.

The metric semantic map m_s, constructed in Sec. 6.2.3, stores the 3D bounding box of $o_{\max}$, which we project to 2D bounding box $\hat{\mathbf{b}}_{\text{2D}}^{\max}$. The shape similarity score is computed from the IoU of the 2D bounding boxes $\hat{\mathbf{b}}_{\text{2D}}$ and $\hat{\mathbf{b}}_{\text{2D}}^{\max}$:

$$p_g(\mathbf{z} \mid m_g, \mathbf{x}_t) = \exp\left(-\left(1 - \text{IoU}\left(\hat{\mathbf{b}}_{\text{2D}}, \hat{\mathbf{b}}_{\text{2D}}^{\max}\right)\right)\right). \tag{6.12}$$

The probability of detecting an object from particle state x_t is given by

$$p(\mathbf{z} \mid m, \mathbf{x}_t) = p_o\, p_g + (1 - p_o)\, \eta, \tag{6.13}$$

where η is an empirically computed constant, representing the weight of a false data association, and p_o and p_g are defined in Eqs. 6.11 and 6.12, respectively. When K objects are detected in a single frame, we compute the overall particle weight as a geometric average

$$p(\mathbf{z}_t \mid m, \mathbf{x}_t) = \prod_{k=1}^{K} p(\mathbf{z}_t^k \mid m, \mathbf{x}_t)^{\frac{1}{K}}. \tag{6.14}$$

An illustration of the localization approach is brought in Fig. 6.7.

6.3 Experimental Evaluation

We present our experiments to show the capabilities of our method. The results of our experiments also support our key claims, which are: our approach is able to (i) generate 3D labels for

Table 6.2. Computed metrics for the two constructed maps compared to the map obtained with a FARO 3D scan. KP was constructed based on known poses from infrastructure, and ICP was constructed with poses extracted from 2D LiDAR ICP.

	map	board	cabinet	desk	drawers	fire ext.	oven	plant	sink	sofa	table	AVG
IoU	KP	0.54	0.75	0.79	0.39	0.55	0.85	0.61	0.77	0.68	0.76	0.67
Pr	KP	1.00	1.00	1.00	0.56	1.00	1.00	1.00	1.00	1.00	1.00	0.96
Rc	KP	0.94	0.92	1.00	0.62	1.00	1.00	1.00	1.00	1.00	0.95	0.94
IoU	ICP	0.51	0.70	0.77	0.40	0.66	0.68	0.60	0.65	0.77	0.75	0.65
Pr	ICP	1.00	1.00	1.00	0.56	1.00	1.00	1.00	1.00	1.00	1.00	0.96
Rc	ICP	1.00	0.92	1.00	0.62	1.00	1.00	1.00	1.00	1.00	0.95	0.95

fine-tuning of 3D object detection models, (ii) enrich floor plans with object information, (iii) and localize in such maps in an online fashion, using onboard computers.

6.3.1 Experimental Setup

To assess the performance of our approach, we made multiple recordings in our building. Our data collection platform was a Kuka YouBot with 2 Hokuyo UTM-30LX LiDARs, wheel encoders, 4 cameras with a joint coverage of 360° field-of-view, and an up-looking camera used strictly for evaluation purposes. The recordings span across 9 months, capturing changes to the lab furniture, varying amount of clutter, human movement and opening and closing of doors.

To extract precise ground truth information about the robot's pose, we use an external localization infrastructure based on densely placed (approx. $1\,\text{tag}/\text{m}^2$) AprilTags, covering the ceiling of each room and corridor of our lab. In every frame captured with the up-looking camera, we detect multiple AprilTags computing the pose estimation in a least-squares fashion and achieving accuracy of under 3 cm. A high resolution point cloud of the lab, generated with a terrestrial laser scanner, was also used to produce the 3D labels used in 6.2.1, and the localization infrastructure was used to generate the poses for the RGB frames used the metric-semantic mapping in 6.2.3.

For Cube R-CNN [16], we fine-tuned the ResNet34-based model [51] the authors provide for indoor perception. For doing this, we created a list of objects of interest suitable for long-term localization. The success of our approach depends on the appropriate choice of the classes of interest. It is important to consider the stability of the object classes, as discussed in Zimmerman et al. [171], and their observability and map coverage. We optimize our model with stochastic gradient descent for 100 epochs, with an initial learning rate of 0.0015 and a batch size of 12. For training, we recorded sequences T1-T7 on the second floor of our building.

As baseline, we compare against HSMCL [171], another semantic MCL framework. All experiments were executed with 5000 particles. The parameters for the different approaches are reported in Tab. 6.1. We consider three metrics, 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 m radius of the ground truth pose, and the orientation is within $\frac{\pi}{4}$. After globally localizing, the tracked pose must not diverge for an accumulated 1.5 s. A localization run is successful if convergence is achieved in the first 95% of the sequence time and the tracked pose does not diverge. Each sequence was evaluated multiple times to account for the inherent stochasticity of the MCL framework, and the success rate is computed over multiple runs.

Figure 6.8. 2D projection for the 3 maps. Left: ground truth map obtained with terrestrial laser scanner. Middle: KP map. Right: map built using scan matching (ICP).

6.3.2 Mapping

To support our second claim, we evaluate the quality of our mapping pipeline. We use sequence M1, which include ground truth poses, RGB stream, and 2D LiDAR scans, while the robot was traversing the entire second floor. We evaluate our constructed maps by matching them against a ground truth (GT) map, extracted with a highly accurate 3D terrestrial laser scanner, and annotated manually. The creation of the ground truth map is labour-intense offline procedure that requires specialized sensors and equipment, providing sub-centimeters accuracy. It has been carried out for evaluation purposes only. Our constructed maps are built using the metric-semantic mapping pipeline describe in Sec. 6.2.3, using data for sequence M1. For the first constructed map, we used precise poses extracted from our localization infrastructure, and we refer to it as known poses (KP) map. We also generated a map using sensor-based poses, estimated via ICP on 2D LiDAR scans. We aligned the estimated poses to an existing grid map, such as floor plan, by initializing the ICP with a known pose. We believe the poses could also be extracted with low-drift visual-inertial odometry pipelines [122]. We evaluated the accuracy of the generated maps directly by comparing it to the ground truth map, considering the IoU between objects in the object-based map, and precision and recall. As can be seen in Tab. 6.2 and Fig. 6.8, the constructed maps are well-matched to the ground truth map: the IoU scores between the ground truth map and our generated map are high, while precision and recall are close to 1, indicating a low number of false positives and false negatives. We also evaluate the performance of robot localization based on the constructed maps. The localization accuracy for the KP map and the ICP map is on par with the accuracy for the ground truth (GT) map, see Tab. 6.3, suggesting that our maps are well-suited for localization.

Table 6.3. Evaluation of the map construction quality through long-term localization performance. The success rate for all maps on all reported sequences is 100%. We report ATE in [rad/m] format for 10 sequences recorded all across our lab in the span of nine months.

Method	Map	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	AVG
Ours	GT	0.089/0.14	0.065/0.11	0.070/0.14	0.043/0.17	0.065/0.16	0.075/0.15	0.073/0.17	0.077/0.20	0.060/0.17	0.047/0.12	0.066/0.15
Ours	KP	0.082/0.14	0.098/0.16	0.082/0.15	0.041/0.14	0.088/0.22	0.075/0.14	0.073/0.16	0.085/0.23	0.065/0.15	0.042/0.14	0.073/0.16
Ours	ICP	0.078/0.14	0.115/0.11	0.085/0.16	0.039/0.15	0.077/0.18	0.081/0.14	0.074/0.18	0.083/0.27	0.057/0.12	0.045/0.11	0.073/0.16

Table 6.4. Baseline comparison for long-term localization on the ground truth map. We report success rate and ATE in [rad/m] format for 10 sequences recorded all across our lab in the span of nine months.

Method	Map	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	AVG
HSMCL	GT	0%	60%	100%	100%	100%	0%	100%	100%	100%	20%	68%
EDT-MCL	GT	100%	80%	100%	100%	100%	100%	100%	100%	100%	0%	88%
D-MCL	GT	100%	60%	100%	100%	100%	100%	100%	100%	80%	0%	84%
O-MCL	GT	100%	80%	100%	100%	40%	0%	100%	100%	100%	0%	72%
Ours	GT	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
HSMCL	GT	-/-	-/-	0.101/0.30	0.147/0.32	0.084/0.25	-/-	0.052/0.19	0.063/0.24	0.078/0.30	-/-	0.088/0.27
EDT-MCL	GT	0.094/0.17	-/-	0.075/0.18	0.043/0.20	0.122/0.16	0.076/0.20	0.073/0.25	0.058/0.23	0.060/0.16	-/-	0.075/0.19
D-MCL	GT	0.068/0.13	-/-	0.055/0.18	0.036/0.17	0.075/0.16	0.037/0.21	0.046/0.24	0.039/0.22	-/-	-/-	0.051/0.19
O-MCL	GT	0.082/0.14	-/-	0.054/0.15	0.045/0.16	-/-	-/-	0.056/0.18	0.049/0.19	0.061/0.14	-/-	0.058/0.16
Ours	GT	0.089/0.14	0.065/0.11	0.070/0.14	0.043/0.17	0.065/0.16	0.075/0.15	0.073/0.17	0.077/0.20	0.060/0.17	0.047/0.12	0.066/0.15

6.3.3 Long-Term Localization in CAD Floor Plans

To assess the performance of our localization approach, we recorded sequences R1-R10, through our building, spanning across nine months. These sequences include the addition and removal of furniture, dynamic obstacles, and quasi-static changes such as the closing and opening of doors. The starting points for the localization sequences were spread between different rooms and the corridor. Accuracy and success rate are reported in Tab. 6.3 for the three different maps. Each map was generated once, and has not been updated during the evaluation period. Convergence time averaged over sequences R1-R10 is 10.9 s for ground truth map and 12.1 s for KP map.

6.3.4 Baseline Comparisons for Semantic Localization

We compare our approach to three strategies for integrating 3D object information into MCL. The first baseline, called EDT-MCL, extends the commonly-used beam-end point model. Based on the semantic metric map m_s (see Sec. 6.2.3), we create an Euclidean distance transform (EDT) for each semantic class. The second approach, D-MCL, computes the particle weight based on the object probability map m_p and the likelihood p_o defined in Eq. (6.11). The third approach, O-MCL, is based on the overlap between a predicted bounding box and the semantic-metric map m_s. The weight is computed based on the overlap score described in Eq. (6.12).

As can be seen in Tab. 6.4, our approach outperforms the baselines. Both EDT-MCL and O-MCL do not incorporate information about the model noise. D-MCL makes use of the learned statistical information about the model performance, but only considers the center of the prediction, discarding valuable information about the object dimensions and rotation. Our approach leverages both the geometric information from the 3D bounding box and takes into account the characteristic model noise, which results in improved performance. Unlike HSMCL, our approach does not use LiDAR information, yet the localization is more robust.

6.3.5 Runtime

Our mapping and localization approaches both run online, onboard of a mobile platform. On our robots, we use an Intel NUC10i7FNK and a NVidia Jetson Xavier AGX. The 3D object detection runs at ∼9 Hz on the NVidia Jetson, and the sensor model executes at 60 Hz on the NUC10i7FNK. This performance is sufficient to construct a 3D metric semantic map and globally localize on our mobile platform. A longer video, including live demos for both mapping and localization, and the code can be found on our GitHub repository at `https://github.com/PRBonn/SIMP`.

6.4 Conclusion

In this chapter, we presented a novel approach for semantic global localization with a complementary 3D mapping procedure to build the object-based maps used for localization. The augmentation of floor plan maps with the 3D metric semantic maps assists navigation in cluttered and dynamic indoor environments. We show that our semantically-guided localization is reliable and accurate on both, the ground truth map and the map acquired with our proposed pipeline, benchmarking it over a dataset of challenging scenarios spanning over nine months. We compared our approach to similar approaches, and the experiments show how long-term localization can benefit from 3D metric semantic maps.

Chapter 7

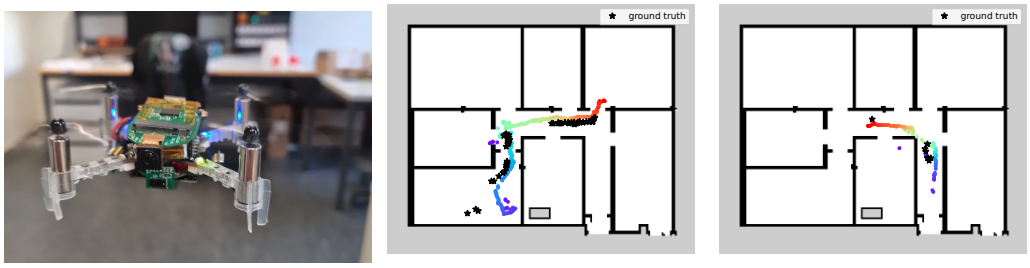
Localization under Resource-constraints

So far we considered localization for standard-sized robots, where power and compute resources impose little to no limitations. For standard-sized robots, RTK-GPS is often used for outdoor localization [132], as well as heavy high-end power-hungry 3D LiDARs [22, 33], which require expensive computations. However, these approaches are unsuitable for nano-UAVs. First, nano-UAVs are usually deployed indoors, in GPS-denied environments. Second, their low payload and limited battery capacity restrict the type and accuracy of sensors and the computational resources available onboard. While deploying algorithms onboard nano-UAVs are challenging, unmanned aerial vehicles (UAVs) are emerging for applications such as monitoring, inspection, surveillance, transportation and logistics [135]. Mainly in indoor scenarios, a small form factor brings key advantages since smaller UAVs allow for safe operation near humans and can reach locations which are inaccessible with larger platforms [48].

One approach to localizing small-sized UAVs in GPS-denied environment is infrastructure-based localization, commonly implemented using ultra-wideband (UWB) [149, 102] or other wireless communication protocols [76, 24]. To tackle the sensing and compute constraints on nano-UAVs, off-board processing was proposed for pose estimation [25, 86]. However, these methods are unsuitable for many application scenarios, as they require prior installation of external infrastructure and reliable communication between the mobile agent and the base stations.

Map-based approaches for localization do not require pre-existing infrastructure or external localization cues, and can therefore operate independently in indoor environments even when communication is unavailable. In map-based approaches, the agent uses measurements acquired by its sensors (LiDAR, camera, sonar, etc.) to estimate its pose in a given map, such as a landmark-based map [9, 47] or an occupancy grid map [94, 32].

Range sensors have been successfully coupled with occupancy grid maps [28, 145] for indoor localization. Unfortunately, these sensors are power-hungry and large and therefore unsuitable for use on a nano-UAV (Fig. 7.1), where only around 10% (around 1-2 Watts) of the overall power budget can be spent on sensing and processing without affecting the flight time substantially [34]. Miniaturized time-of-flight (ToF) sensors [97] were used to localize a nano-UAV. However, they suffer from a short range and low beam count (8×8) that limits their ability to operate in large spaces.



(a) (b) (c)

Figure 7.1. (a) A nano-UAV while flying and globally localizing in an office environment using our novel sensor fusion approach. (b-c) A qualitative evaluation of the localization results on recorded sequences. Ground truth pose is marked by black stars. The rainbow colors encode the time of prediction, with purple marking the beginning of the sequence and red its end.

Sparse maps, such as floor plans, are attractive due to their availability, removing the need for a complex mapping procedure before deployment. However, relying solely on geometric information from range sensors can lead to global localization failures in sparse maps and environments with high structural symmetries [171].

Humans navigate using objects rather than precise metric measurements [84, 166], which inspires leveraging semantic information to improve localization. With the recent advances in tasks such as object detection [12], semantic cues are commonly utilized for robot localization [2, 166, 171]. However, they still require high-end, energy-consuming computational resources, which are usually not available on nano-UAVs. The challenge of executing semantic inference under limited computational resources is addressed with neural architecture search, quantization and optimized deployment engines [35, 87, 156].

Our main contribution is an approach for global indoor localization on a nano-UAV. In our approach, we use sensor fusion to exploit both semantic and geometric information. We fully execute our online and onboard approach on a novel low-power processor. We address the difficulty of executing object detection tasks on resource-constrained platforms, describing a pipeline for model quantization and deployment. We introduce a memory-efficient map representation which contains both semantic and geometric information. We utilize semantic information extracted from the camera and fuse it with range measurements from a miniaturized ToF sensor to localize a nano-UAV in a the metric-semantic map, by introducing a novel observation model, as seen in Fig. 7.1.

In our evaluation, we demonstrate that our approach can (i) globally localize a nano-UAV in a given map, (ii) infer semantic cues under resource constraints, (iii) operate with low power consumption onboard (iv) execute in real-time (15 Hz ToF, 5 Hz camera). Additionally, we provide a video of the live demo, as well as an open-source implementation.

The research presented in this chapter covers two published works. The first, titled "**Fully On-board Low-Power Localization with Multizone Time-of-Flight Sensors on Nano-UAVs**" [97], was accepted to *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2023. The second, titled "**Fully Onboard Low-Power Localization with Semantic Sensor Fusion on a Nano-UAV using Floor Plans**" [174], was accepted to *IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2024.

7.1 Related Work

Recent literature has demonstrated that sensing and processing on nano-UAVs are strongly limited; therefore, many previous works have proposed solutions for autonomous navigation that only rely on simple state estimation techniques such as an inertial measurement unit and odometry for localization [83, 96, 109, 101]. The major drawback of these approaches is their inability to compensate for drift and recover from accumulated errors [96]. Most drift-correction approaches use range measurements to anchors with known locations [102].

In indoor scenarios, the most commonly used technology is UWB, but approaches with Bluetooth or WiFi are also available. They all have disadvantages — they require line-of-sight between nodes, depend on pre-installed infrastructure [57, 102] or can only estimate relative position [149]. The resulting mean localization errors are often over 20 cm (22 cm in [102], 28 cm in [149]). Coppola et al. [24] present a Bluetooth based relative localization approach where signal strength is used as a range measurement. In contrast to previous works, our approach focuses on an infrastructure-less approach to globally navigate indoors: a map-based localization approach using particle filters, which was not explored on nano-UAVs until now. Probabilistic approaches provide robust localization and include seminal works such as the extended Kalman filter (EKF) [68], Markov localization [39] and particle filters often referred to as MCL [28]. These approaches are suitable for localization using range sensors such as 2D LiDARs, sonars, and cameras. Until now, these approaches were nearly infeasible on nano-UAVs, due to bulky power-hungry sensors, and high computational demands, which are hard to satisfy on embedded systems.

For both the sensing and the processing challenges, promising hardware recently emerged. Although introduced on the market only recently, lightweight multizone ToF sensors are already working well for obstacle avoidance [96]. As for powerful and energy-efficient SoCs for processing, SoCs of the parallel ultra-low power (PULP) family have been employed on drones before. The GAP8 SoC is utilized for corridor [101] or person following [109]. These approaches use deep learning with quantized models but do not venture into float-heavy tasks such as particle-filter localization. A novel SoC, GAP9[1] was recently released, which with 0.33 mW per giga operation (GOP) is an order of magnitude more power efficient than GAP8 and most importantly, features increased memory and floating point support.

Localization in indoor human-oriented environments is particularly challenging, due to the presence of dynamic obstacles such as humans, chairs and carts, as well as quasi-static changes such as opening and closing of doors and rearrangement of furniture [172]. Relying solely on geometric features may lead to global localization failure, especially when localizing on sparse maps such as floor plans [171]. Additional sources of information, such as WiFi and textual cues, have been integrated into localization frameworks [26, 57, 170] to increase the robustness of localization, as well as semantic information about objects in the scene Mendez et al. [84], Zimmerman et al. [171].

In our previous work [171], we propose a semantic localization utilizing both 2D laser and a camera, and our current work shares the concept of abstract semantic map representation. Unlike the previous method which requires a 360^o LiDAR and camera coverage, and a power-hungry onboard computer (Intel NUC10), we demonstrate semantic localization with less than 1% of that power consumption and sensors that are orders of magnitude smaller.

Extracting semantic information is essential for semantically-guided localization. While

[1] `https://greenwaves-technologies.com`

lightweight architectures such as YOLO [12] enable inference of object detection models onboard computers such as Intel NUC and Nvidia Jetson, they still require several tens of megabytes of memory for sensor-rate execution and are therefore unsuitable for execution on microcontrollers(MCU). Recent years witness a growing interest in the deployment of machine learning on edge devices, specifically semantic perception tasks such as object detection [56, 74, 92]. Motivated by these trends, our approach incorporates also semantic information from the onboard camera, to overcome the range limitation of the ToF sensors.

In this chapter, we combine a miniature multizone ToF sensor with a novel processor to enable on-board infrastructure-less localization in indoor environments with an accuracy that surpasses the state of the art of localization in nano-UAVs with UWB [149, 102]. We first introduce our optimized ToF-based MCL implementation [97]. We discuss its limitations, as it cannot operate in larger, open environments due to the short range of the ToF sensors (2.5 m), and propose a sensor-fusion semantic localization approach suitable for nano-UAVs.

To the best of our knowledge, our approach is the first to fuse both range measurements and semantic cues for global localization on nano-UAVs. By optimizing the map format and sensor model for a novel ultra-low-power processor, we are able to globally localize, onboard and online, in indoor environments without the need for infrastructure or reliable communication [102, 149]. Additionally, we are the first to deploy a state-of-the-art (SotA) object detection model from the YOLO family on a RISC-V multi-core platform, achieving 20 Hz with only 2.5 mJ per frame.

7.2 System Overview

We introduce the hardware setup including the nano-UAV, sensors, and compute platform. We use a Crazyflie 2.1, an open-source drone, and extend it with three plug-on decks, as shown in Fig. 7.2. Note that the WiFi module, the 2.4 GHz radio and the uSDcard solely serve for logging and remote steering purposes, no computations are offloaded. For time synchronization of the logging we send a timestamp packet from the STM32 on the Crazyflie to the GAP9 every 10 ms. Fig. 7.2 displays the interactions between the different components introduced below.

7.2.1 Hardware: Crazyflie and Extension Boards

The Crazyflie 2.1 is a commercially available open software/hardware nano-UAV. In this work, we use its inertial measurement unit (IMU), radio communication (using an nRF51822, solely to log data and steer the drone) and the main processor, an STM32F405 (168 MHz, 192 kBRAM), which is responsible for sensor readout, state estimation, and real-time control. We equip the Crazyflie 2.1 with upgrade-kit motors and propellers and a 350mAh battery.

Flow-deck v2

a commercially available deck featuring a downward-facing optical flow sensor and 1D ToF sensor for odometry measurements. Those sensors improve the internal position estimate provided by the Crazyflie firmware through an extended Kalman Filter.

Multizone-ToF-deck

a custom deck featuring up to 4 VL53L5CX sensors, which can provide a matrix of either 8x8 or 4x4 pixels at maximally 15 Hz or 60 Hz respectively. For each zone, it provides a distance measurement coupled with an error flag, which gets raised when out-of-range measurements or interference are detected. Each sensor has a 67° diagonal field of view (FoV) and a maximal range of ∼2.5m [41].

GAP9-deck

a custom deck, featuring GAP9, which integrates an RGB camera (OV5647) and a RISC-V parallel system-on-chip called GAP9[2] as processing platform. It also includes a NINA WiFi module, which is used for data collection. In our application, the multizone ToF sensor measurements are acquired by the STM32 via an I2C bus and then, together with the state estimation, sent via SPI to the GAP9 SoC.

7.2.2 Processor: GAP9

GAP9's architecture is based on the open-source SoC Vega [123] and features 10 RISC-V instruction set architecture-based cores, extended with custom instructions. The compute cluster, featuring 9 cores, one for orchestration and 8 workers, delivers programmable compute power at extreme energy efficiency. GAP9 features 128 kB of shared L1 memory. The fabric controller (FC) has access to various peripherals and features 64 kB RAM, 1.5 MB interleaved memory (L2) and even 2 MB flash. The architecture employs adjustable dynamic frequency and voltage domains, allowing us to tune the energy consumption to the exact requirements at a particular point in time. At peak performance, the cores run at 400 MHz on both the cluster and the FC. GAP9 also features NE16, a convolutional neural network (CNN) hardware accelerator, specialized for highly efficient MAC operations. NE16 is tailored to 3×3 convolutions, as it features 9×9×16 8x1bit MAC units, but it also offers support for 1×1 and 3×3 depth-wise convolutions and fully connected layers. Additionally to the aforementioned internal memory on GAP9, we mount an L3 RAM octa-SPI memory of 32 MB. As our approach requires a camera interface and the execution of multiple computationally heavy loads with relatively high memory requirements (>1 MB), making GAP9 a good fit.

7.3 Approach

We aim to globally localize a nano-UAV in a given floor plan, targeting full-scale human oriented environments, such as offices. To this end, we fuse geometric and semantic information, in a MCL framework. In Sec. 7.3.1 we detail the training and deployment pipeline for porting a SotA object detection model to a resource-constrained MCU. We briefly introduce our lightweight, parallel MCL implementation in Sec. 7.3.2. In Sec. 7.3.3 we describe our optimized semantic map format, and then present our novel fusion sensor model in Sec. 7.3.4.

[2] https://greenwaves-technologies.com/gap9_processor

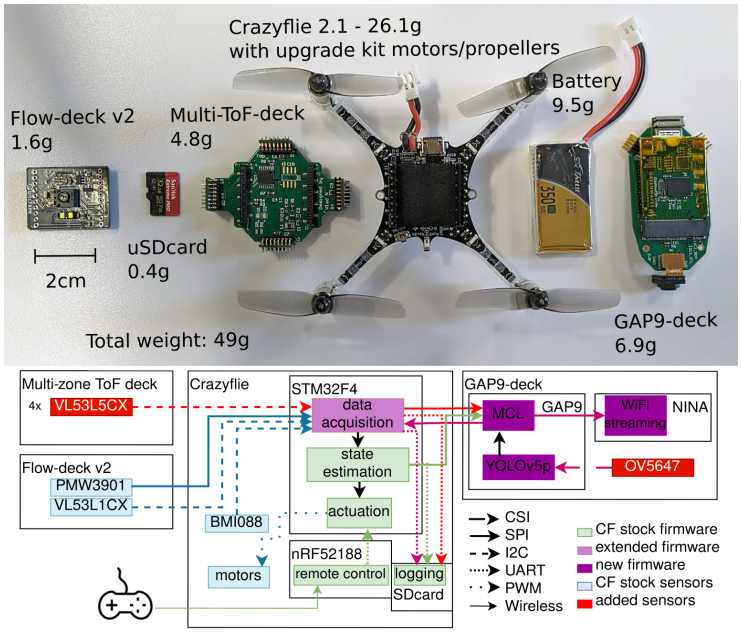


Figure 7.2. System overview. Top: All stacked components on the nano-UAV, ordered from the top (left) to the bottom (right). Bottom: A visualization of the communication paths and task distribution between all employed processors and sensors.

7.3.1 Object Detection

We use a modified architecture from the YOLOv5 family, we refer to as YOLOv5p, to reduce the memory and execution time required for inference fully onboard, on the parallel RISC-V processor. The proposed YOLOv5p has a smaller backbone and also a reduced head, with 623K parameters compared to the 1.9M parameters of smallest YOLOv5 model(YOLOv5n). We first pre-train our model on the COCO dataset [75] for 100 epochs, and then fine-tune on our custom dataset, learning semantic classes of interest. To deploy the model on the GAP9, we use the deployment pipeline NNTool[3], which includes quantization, an inference engine for verification in python and code generation for deployment on resource-constraint MCUs. It supports the NE16, a hardware accelerator present in GAP9.

7.3.2 Lightweight and Parallel Embedded Implementation of MCL

The two main constraints in the nano-UAVs hardware are memory and execution time, and both resources must be used efficiently to enable MCL on-board in real-time. Our implementation of MCL is asynchronous – the motion model is sampled when odometry is available, and the particles are re-weighted when new range measurements arrive. We only consider new observations if the drone moves more than d_{xy} or rotates more than d_θ. However, we configured our sampling rates for the motion and observation update to be the same, limited by the 15 Hz maximum update rate of the ToF sensor.

[3] https://github.com/GreenWaves-Technologies/ gap_sdk/tree/master/tools/nntool

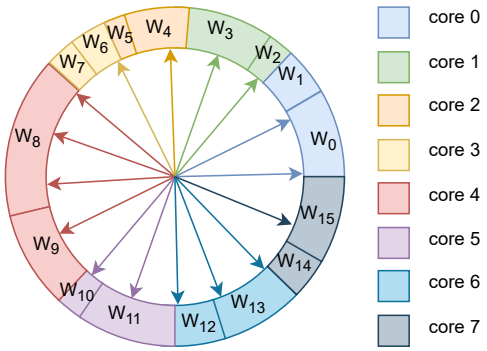


Figure 7.3. Parallelizing the resampling wheel: Each color represents a core, the current particles are distributed evenly (here two per core) and then the new particles are chosen according to where the arrows of the resampling wheel point.

The motion model, observation model and pose computation can be parallelized exploiting the GAP9 cluster by distributing the particles among the cores. The resampling step can also be parallelized, but due to its dependency on all weights, the workload distribution cannot be planned optimally. The first step is weight normalization, which involves computing the sum and dividing by it – we can parallelize this step by splitting the particles evenly to all cores. We also store the partial sums, as we can use them to parallelize the main resampling step, drawing the new particles. For drawing the new particles, we use a systematic resampling algorithm [29], which we explain with the analogy of a wheel, as shown in Fig. 7.3. We draw one random number, corresponding to the first *arrow* in the wheel, with the other $N-1$ *arrows* being fixed at regular intervals from that randomly picked *arrow*. The colors show how we distribute the drawing of the next particles to the cores. The current particle weights are represented by the cone area they occupy. As we know the partial sums computed by all cores, we can directly use them to calculate which core will resample how many particles and which ones. In Fig. 7.3, the colored *arrows* represent the new particles picked by the corresponding cores.

The main components of MCL using memory space are the particles and the map. The occupancy map requires just 2 bits per cell (to represent the 3 possible states). However, we also precompute the EDT values for each cell, leading to an additional floating point number being saved for every cell. To decrease the memory usage, we compare three possibilities: 32-bit floating point numbers, 16-bit floating point numbers, and quantized 8-bit unsigned integer values. Each particle stores 4 components, (x, y, θ, w). With a 32-bit floating point representation, this leads to 16 bytes per particle. However, as we are double-buffering the particles for executing the resampling step, we need 32 bytes per particle for the 32-bit representation, and 16 bytes for more memory-efficient 16-bit representations.

7.3.3 Semantic Map Format

The proposed semantic map format contains geometric information, in the form of an occupancy grid map extracted from a floor plan, which is enhanced with prior semantic information. Similarly to our previous work [171], we choose a simplified representation for our semantic information, defining objects by their semantic class and a 2D bounding box. However, the

contribution of our work is a unified, memory-efficient map representation, instead of multiple semantic visibility maps. In the proposed approach, the occupancy states (free, occupied, unknown) are represented by 2 bits, and the semantic maps are represented by 1 bit per class, resulting in one 16-bit map. This generic representation of semantic objects enables fast, manual annotation and removes the need for an expensive mapping procedure. Our approach can handle inaccurate annotations of both object pose and size. A colored visualization of the semantic maps can be seen in Fig. 7.4.

7.3.4 Geometric-Semantic Fusion Sensor Model

The output of our object detection model contains the class label, the bounding boxes coordinates in a $xyxy$ format and the confidence score for the detected object. First, we compute the center of the bounding box $\mathbf{v}_c = (x_c, y_c, 1)$ in homogeneous coordinates and project it to a 3D ray in the camera frame

$$\mathbf{V}_c(\lambda) = \mathbf{O} + \lambda \mathbf{R}^{-1}\mathbf{K}^{-1}\mathbf{v}_c, \tag{7.1}$$

where $\mathbf{K} \in \mathbb{R}^{3\times3}$ is the camera intrinsics, $\mathbf{R} \in \mathbb{R}^{3\times3}$ is the camera rotation and $\mathbf{O} \in \mathbb{R}^3$ is camera center. As the camera is aligned to the forward direction of the nano-UAV, we assume the rotation matrix is unity. For every particle $s_t^i \in \mathcal{S}$, we transform the 3D ray $\mathbf{V}_c(\lambda)$ to the map coordinate frame using the pose $\mathbf{x}_t^i$. Then we trace the ray in the semantic map, from $\mathbf{x}_t^i$ in the direction of the transformed ray $\mathbf{V}_c^m(\lambda)$. We consider the tracing to fail if we encounter an occupied cell before reaching an object of class c, and in this case we penalize the particle by lowering its weight to w_{penalty}. As the FoV of the front ToF sensor and the camera overlap, we can associate range measurements to detected objects. If the tracing was successful, we look up the 8×8 ToF beam measurements corresponding to the bounding box, and calculated the average distance to the object d_{ToF}. If the distance is less than τ_t, we match the traced distance d_{trace} to the measured distance d_{ToF}

$$p_s(\mathbf{z}_t \mid m, \mathbf{x}_t) = \frac{1}{\sqrt{2\pi\sigma_s}} \exp\left(-\frac{(d_{\text{trace}} - d_{\text{ToF}})^2}{2\sigma_s^2}\right), \tag{7.2}$$

where $\mathbf{z}_t$ includes both the semantic observation inferred by our object detection model and range measurements from the front ToF sensor. When semantic information is not available, we use the ToF observations to re-weight the particles according to the Beam End Model [146](Sec. 2.1), with 32 range measurements, extracting 8 beam points from the middle row of our 4 ToF sensors. To reduce memory consumption, the EDT was truncated at distance r_{max} and quantized to 8-bit for memory efficiency. As the ToF measurements are unreliable after 3 m, we discard observations beyond that range, and only perform the update step with enough valid measurements. We employed an aggressive resampling strategy, sampling the particle after every observation.

7.4 Experimental Evaluation

We evaluate the proposed method in several experiments, to support our claims the we can (i) globally localize a nano-UAV in a given map, (ii) infer semantic information under resource constraints, (iii) operate with low power consumption onboard, (iv) execute in real-time (15 Hz

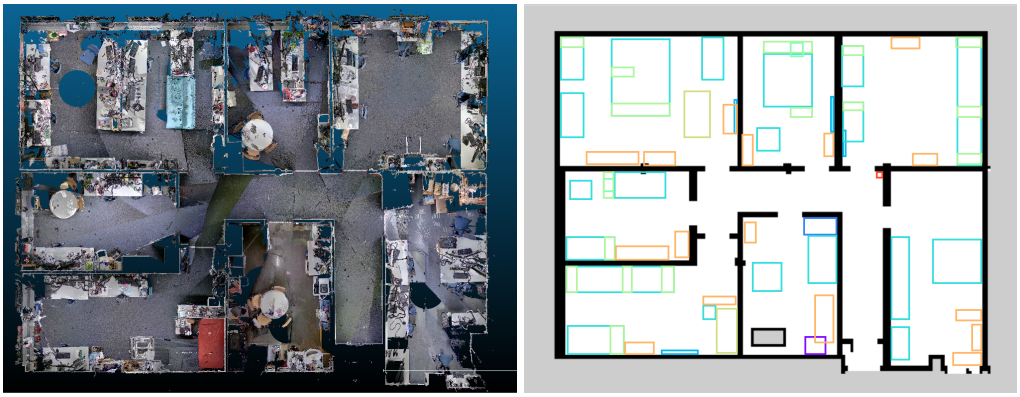


Figure 7.4. Left: A top view of the dense pointcloud captured with the Z+F Imager 5016 terrestrial laser scanner, which was used solely for GT extraction. The full pointcloud has 200 million points. Right: The semantically-enriched floor plan of the lab. Semantic objects of interest are represented using their bounding box and class ID. Different colors represent different object classes. The semantic information was added manually, without a complex measuring or mapping procedure.

ToF, 5 Hz camera). Specifically, we show that we can localize in a featureless map such as a floor plan using a low-power compute platform and miniaturized sensors, due to leveraging semantic information.

7.4.1 Experimental Setup

To evaluate our approach, we recorded 10 piloted drone flights (S1-S10) over several weeks, spanning across the lab (Fig. 7.4), including quasi-static changes, furniture moving and different lighting conditions. The recordings include odometry from the Crazyflie's internal state estimation and ToF measurements at 15 Hz. For the front ToF sensor, we recorded the full 8×8 grid, while for the right, back and left ToF sensors, we only recorded 8 beams extracted from the middle row of the grid, due to bandwidth limitations. We also recorded images with 640×480 pixel at a lower rate of 2 Hz, due to Wi-Fi streaming limitations.

To assess the accuracy and robustness of our approach, we used AprilTags [104] to compute the pose of the nano-UAV. We placed 148 AprilTags on the walls, and then used a SotA laser scanner to create a dense 3D pointcloud of the lab, shown in Fig. 7.4. The 3D coordinates of the AprilTags were extracted by running the detector on orthographic projections of the walls. In images where the AprilTags are detectable, we used 2D-3D correspondences to estimate the nano-UAV's position [80]. The images were recorded at 640×480 pixel to enable the detection of AprilTags. Due to the low resolution and the blurry nature of in-flight images, the GT accuracy is ~ 0.1 m. The AprilTags are used strictly for evaluation and are not part of the localization approach. In addition, we collected training, validation and test sets to train and benchmark our object detection model.

The given map (Fig. 7.4) is a floor plan augmented with semantic information in the form of bounding boxes, annotated by a lab member from memory. The semantic annotations are imprecise, and did not require any type of measurement, but are simply hand-drawn. In our abstract map, objects differed from their actual size by 50%, or up to 1 m. The map resolution is 0.05 m/pixel, covering an area of 280 m^2.

Table 7.1. Average precision(AP) (IoU=0.50) scores for the test set, confidence TH 0.2, IoU TH 0.5

Class	sink	door	fridge	board	table	plant	drawers	sofa	cabinet	extinguisher	all
YOLOv5p	0.663	0.579	0.952	0.574	0.51	0.379	0.604	1.0	0.826	0.967	0.705
FP32	0.663	0.508	1.0	0.525	0.515	0.337	0.659	1.0	0.723	1.0	0.693
MIXED	0.663	0.482	1.0	0.752	0.516	0.168	0.554	1.0	0.715	1.0	0.685
UINT8	0.663	0.492	1.0	0.644	0.436	0.168	0.604	1.0	0.68	0.851	0.654

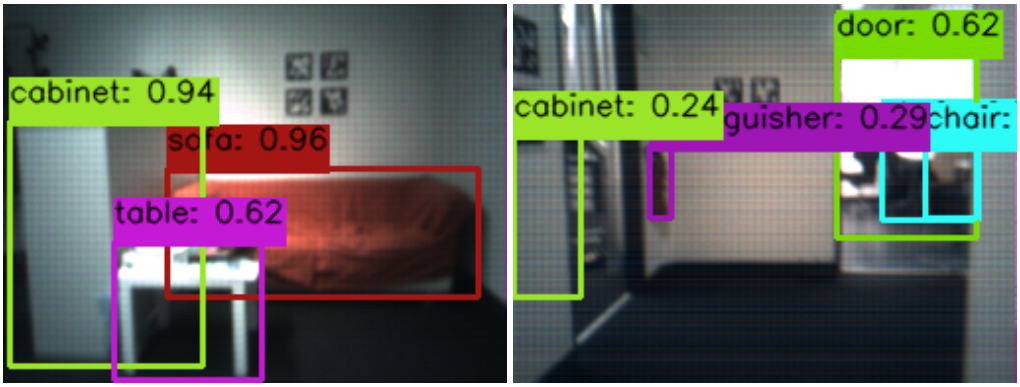


Figure 7.5. A qualitative evaluation of the 8-bit quantized object detection model on 256 × 192 input images.

7.4.2 Object Detection Performance

We evaluated the quantized YOLOv5p object detection model to ensure that our deployment process can preserve the accuracy of the full precision model. While images are acquired at 640x480 pixel, they are downsampled to 256 × 192 for inference. We compared the average precision for each class of interest and the mean average precision for 4 variations. We trained YOLOv5p using the Ultralytics framework [59]. FP32 is a full precision model converted by the NNTool from YOLOv5p. MIXED is a quantized model with varying precision - the first layer is FP16, and the rest are UINT8. UINT8 is a NNTool 8-bit quantization of YOLOv5p. YOLOv5p inference was executed on a NVidia GTX3070 GPU. MIXED and UINT8 inferences were executed on the GAP9. For the evaluation, we collected a test set with 50 images, including several instances of each class of interest. The images were captured by the nano-UAV's onboard camera. As can be seen in Tab. 7.1, we lose up to 7.2% accuracy in the quantization and deployment process, but the performance is still satisfactory, enabling the extraction of semantic information for localization. Inference examples from the UINT8 model can be seen in Fig. 7.5.

Our object detection pipeline consists out of four parts: (i) image acquisition (ii) preprocessing (iii) quantized neural network (iv) post-processing. As we experienced limitations in the camera acquisition speed it takes 50ms to acquire an image. As the image is acquired by the μDMA [123], with double buffering this time can be used productively on GAP9. The second part is preprocessing, which includes demosaicing and transforming from 10 to 8 bit inputs and takes 5ms. The quantized neural network takes 38 ms, and the post-processing (non-maximum suppression) takes 0.3 ms. This means that the limiting factor is the image acquisition - currently we can reach 20 Hz, however, even if the hardware would allow to acquire images faster 23 Hz would still be the upper limit for the UINT8 network.

Table 7.2. Algorithm parameters

Method	σ_{odom}	σ_g	σ_s	τ_t	$r_{\max}$	d_{xy}	d_θ	w_{penalty}
Nano-SMCL	(0.5, 0.5, 0.5)	8.0	10.0	2.5 m	2 m	0.05 m	0.05 rad	-
Nano-MCL	(0.5, 0.5, 0.5)	8.0	-	-	2 m	0.05 m	0.05 rad	0.01

Table 7.3. Evaluation of the approaches on recordings S1-S10 with 4096 particles. Top: Absolute trajectory error in meters. Bottom: convergence time in seconds.

Method	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	AVG
Nano-SMCL	0.47	-	0.30	0.22	0.36	0.22	0.40	0.25	0.29	0.37	0.32
Nano-MCL	-	-	-	-	-	0.26	-	-	-	-	0.26
Nano-SMCL	30.87	-	42.39	101.43	19.30	30.32	56.04	54.78	24.69	46.57	45.15
Nano-MCL	-	-	-	-	-	67.02	-	-	-	-	67.02

7.4.3 Global Localization in Floor Plans

To evaluate the capability of our localization approach, we examined 3 metrics. The success rate, convergence time and absolute trajectory error (ATE). Since our ground truth (GT) positions are not continuous, as AprilTags are not visible or detectable in every frame, we only evaluated our predictions at the timestamps where the GT checkpoints were available. We consider the point of convergence to be when the ATE is lower than 0.5 m. We consider a localization to be successful when the pose estimation remains converged until the end of the sequence. The algorithm parameters are specified in Tab. 7.2. We tested our approach (Nano-SMCL) with 4096 particles, that were initialized uniformly all over the map. For inferring the semantic cues, we used the 8-bit quantized model, with 256×192 input images.

As a baseline, we used the ToF-based MCL approach [97], referred to as Nano-MCL, providing input from 4 ToF sensors and not relying on semantics. As portrayed in Tab. 7.3, our algorithm converges with 90% success rate, with an average ATE of 0.32 m. Our average convergence time is 45 s. A qualitative evaluation of our localization results can be seen in Fig. 7.1. Our approach failed to localize on sequence S7. In this short sequence, the nano-UAV was mostly in the center of a large and cluttered room, and we could not make use of the ToF measurement. In addition, there is also an ambiguity related to the semantic information, where the configuration of sofa, cabinet and table in close proximity also exists in another room, causing the particle filter to maintain two hypotheses as can be see in Fig. 7.6. We outperformed the range-only MCL approach on all criteria, showing that relying solely on geometric information leads to a success rate of only 10% We speculate that Nano-MCL is suitable for more detailed maps such as occupancy grid maps, and also for environments without vast empty spaces. This is a limiting factor for the short-sighted ToF sensor, and motivates the additional use of semantic information.

7.4.4 Real-time execution, power and memory footprint

In this section, we present the execution times, power measurements and memory footprint of our approach.

Execution times In Table 7.4 we summarize the average execution times (FC and cluster

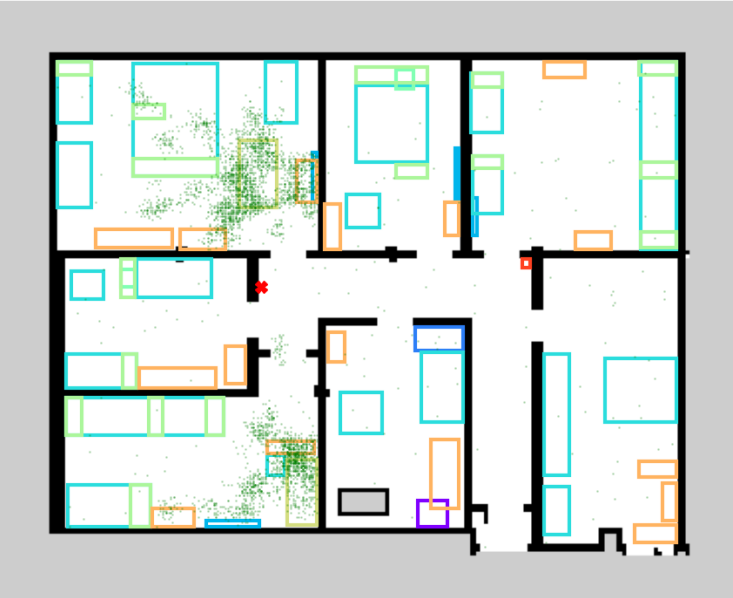


Figure 7.6. A failed localization scenario due to ambiguity in both geometric and semantic features. The particles, marked as green dots, are divided between two rooms with similar properties. The weighted-average prediction is marked with a red cross.

running at 370 MHz) of the different steps, and the resulting processor load per task at the worst-case execution rate, using the maximal 15Hz from the ToF sensor for the ToF and odometry update, as well as 5 Hz for the camera (all tests are executed at around 2 Hz, limited by streaming for debugging and repeatability purposes).

When parallelizing on 8 cores, we reach an overall average speedup of 4.5 on the MCL, leading to a maximum worst-case load of 0.77 and demonstrating that our semantic localization approach can run in real-time. Note that this is a worst-case scenario for computations, since we assume the maximum frequencies we can acquire data with. In practice, the MCL updates are only performed under certain conditions, such as having valid observations and moving a threshold distance. Note that the camera acquisition time does not imply any processor load per se, as it can be executed by the μDMA [123], however, due to camera driver limitations we can only acquire images with the full FoV in VGA resolution and with 10 bits per pixel, saved in 2 bytes, resulting in a too big image to double buffer it in L2. The data coming from the Crazyflie's STM32 (odometry estimation and ToF measurements) is much smaller and can be double buffered.

Power consumption Power consumption of the drone can be divided to 3 main categories: actuation, sensing, and processing, with the motors consuming the most at $\sim 15W$. The ToF sensors require 266mW each in continuous operation, accumulating to 1.06W, while the camera consumes only 90 mW. The Crazyflie stock electronics consume ∼280 mW. The added processor, GAP9, consumes on average 64.7mW during the quantized YOLOv5p execution and 23 mW during the MCL execution, which results in an average processing and sensing power consumption of just under 1.5W, staying inside the 10% power budget usually advised for sensing and computation on nano-UAVs. For comparison, onboard compute platforms for standard-size robots, such as Intel NUC10, consume up to 120W, 5 orders of magnitude more than the GAP9.

Table 7.4. Execution time, worst-case execution rate, and resulting processor load for single- and multi-core implementations (where present).

	camera acqu.	pre-process	NN	post process	cam/ToF fusion	obs. ToF	motion	resampling	**total**
Worst-case exec. rate (Hz)	5	5	5	5	15	15	15	15	
Single-core (ms)	50	4.5	-	0.3	43.9	39.1	10.3	0.6	
Load single-core	0.25	0.02	-	0.00	0.66	0.58	0.15	0.01	**1.67 + CNN**
Multi-core (ms)	-	-	38	-	8.5	8.6	2.6	0.5	
Load multi-core	(0.25)	(0.02)	0.19	(0.00)	0.13	0.13	0.04	0.01	**0.77**

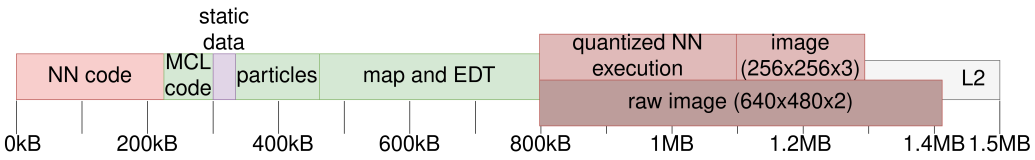


Figure 7.7. The 1.5 MB L2 memory on GAP9 is used for code and data.

Memory The main constraint, fitting everything into the assigned L2 space, which we illustrate in Fig. 7.7 As we update the MCL at up to 15 Hz, we allocate L2 memory for the particles for faster access. This requires 128kB for 4096 particles, where each particle's state $s_t^i = (\mathbf{x}_t^i, w_t^i)$ is represented by 4 floats (32-bit). For the 371x302 pixel semantic map, each pixel is represented by 16-bit value to encode the occupancy grid map and the multiple semantic maps. We also store a quantized 8-bit EDT, and both map and EDT require 336 kB. We acquire raw images in VGA resolution ($640 \times 480 \times 22$bytes), totaling in 614 kB, and preprocess them to 256×256 8-bit RGB images, reducing the memory consumption to 196 kB. The NNTool allows to limit L2 size allocated for the model inference in exchange for slower execution (as transfers from external octa-SPI RAM are necessary). We allocated 300KB for our 8-bit quantized YOLOv5p as the inference time was only 3 ms longer than with 1MB. Additionally, code and static data occupy L2 memory as well - 75KB for the MCL code, 225 kB for the neural network inference and 34 kB for static data. The overall peak memory usage is 1.41 MB out of the available 1.5 MB, enabling the implementation of additional functionalities such as obstacle avoidance or navigation.

7.5 Conclusion

This chapter presents a fully-onboard, global localization approach for a nano-UAV, operating in a full-scale, human-oriented indoor environment. The proposed approach exploits low-element count, miniaturized ToF sensors, fusing the range measurements with semantic information extracted from the onboard camera, to localize in a semantically-enhanced floor plan. We present a SotA object detection model at 20 Hz and 2.5 mJ per frame on a < 100 mW RISC-V multi-core processor. We provide an optimized semantic map format and a sensor model that are suitable for onboard, online execution on nano-UAVs. In our experiments, we show the benefit of exploiting semantic cues for localization, and demonstrate that our approach can successfully localize in various real-world scenarios.

Chapter 8

Collaborative Localization

Previously, we discussed the challenges of localization 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 [60] and Wi-Fi signal strength [57] can be used to improve pose estimation, as well as textual cues [26, 170]. Another venue is utilizing semantic information [2, 53, 60], harnessing the significant progress in the fields of semantic understanding. Despite our best efforts to enable robust long-term single-robot localization, it can still fail. 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 [4, 40, 114]. 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 [114], is computationally costly and has high bandwidth consumption. Therefore, a compressed representation of the belief is imperative and has been proposed in the past [114]. In this work, we analyse the computational complexity related to compressing, exchanging, and fusing robots' beliefs for collaborative localization.

The main contribution of this work is a novel approach to collaborative global localization (Fig. 8.1) 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 multi-robot scenarios, our experiments focus on the case of two collab-

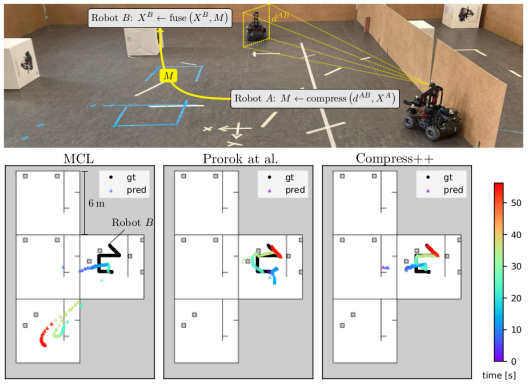


Figure 8.1. Top: two robots during one experimental run when robot A detects robot B. Bottom: localization of robot B in the same run using 3 methods; prediction, color-coded for time, is plotted against ground truth position (black). Two failures, using non-collaborative MCL and Prorok et al. [114], and a successful convergence after 20 s with our collaborative localization approach (Compress++).

orating robots, where we show that our approach (i) improves collaborative localization, (ii) decreases the required bandwidth, (iii) reduces the computational load, (iv) runs online on an onboard computer.

The chapter is organized as follows. Sec. 8.1 provides a summary of related research in collaborative localization. In Sec. 8.2, 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. 8.3. We introduce our experimental setup in Sec. 8.4. In Sec. 8.5, we evaluate the performance of our approach against several baselines, and offer insights regarding their different behaviors.

This work, titled "**Resource-Aware Collaborative Monte Carlo Localization with Distribution Compression**" [173], was accepted for *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2024.

8.1 Related Work

Map-based localization is an integral part of enabling the autonomy of mobile robots [18, 146]. Global localization for a single robot is widely-researched, and probabilistic methods [28, 39, 146] 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 [54, 81], we focus on cooperative global localization in a given map [40].

Multi-centralized approaches [98, 99, 125], 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 [2, 53, 60, 171, 172], leveraging semantic scene understanding [12, 23, 52, 140] was adopted for multi-robot localization. A common approach to collaborative localization relies on robot detection, where one robot can sense another robot. Fox Fox et al. [40] propose a factorial representation where each robot maintains its own be-

lief, 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. [4], propose a system for collaborative localization based on Monte-Carlo localization [28] framework, but do not include an explicit detection model.

Wu and Su [161] present an improvement to Fox et al. [40], 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. [106] 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 and Martinoli [112] 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 [114], they provide a clustering algorithm to reduce the complexity to $O(NK)$, where K is the user-defined number of clusters. In all works by Prorok et al. [114], Prorok and Martinoli [112], 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 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 [40, 114], as well as present resource-aware alternative detection models. We benchmark the various approaches on several environments, and offer insights regarding their performance.

8.2 Approach

We aim to globally localize a team of robots in a given map. We describe in Sec. 8.2.1 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. 8.2.2, we introduce our novel approach and, in Sec. 8.2.3, we give an overview of alternative methods.

8.2.1 Collaborative Monte Carlo Localization

Collaborative Monte Carlo localization extends the single robot MCL (Sec. 1) 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. When robot A detects robot B at time t, it estimates the *relative* position $d_t \in \mathbb{R}^2$ of B and then sends message M to B with information about its state X_t^A and detection d_t, as illustrated in Fig. 8.1.

B, upon receiving the message, updates the weights of its particles [40], fusing information:

$$w_{t,i}^B = w_{t-1,i}^B p\left(\mathbf{x}_{t,i}^B \middle| M\left(d_t, X_t^A\right)\right). \tag{8.1}$$

A straightforward but naive way of implementing the right-most factor in Eq. (8.1) is to include the entire particle set X_t^A in the message. Then

$$p(\mathbf{x}_{t,i}^B | d_t, X_t^A) = \sum_{j=1}^{N} w_{t,j}^A p(\mathbf{x}_{t,i}^B | d_t, \mathbf{x}_{t,j}^A), \tag{8.2}$$

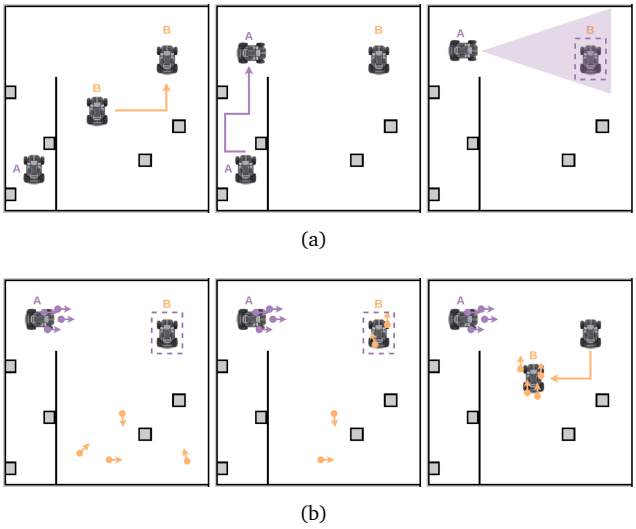


Figure 8.2. (a) An illustration of an experimental run up to the first detection event. (b) A run where robot B has no particles around its truth position at the time of detection (left), followed by reciprocal sampling (center) and successful localization (right).

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

$$p\left(\mathbf{x}^B\middle|d,\mathbf{x}^A\right)=\mathcal{N}\left((x,y)^B;T(d;\mathbf{x}^A),\Sigma\right), \tag{8.3}$$

where $T(\cdot;\mathbf{x}^A)$ transforms a relative position with respect to pose $\mathbf{x}^A$ to an absolute position; please note that detections have no information about orientation. This naive implementation has quadratic time-complexity (see Section 8.3), which is problematic when resources are limited. For this reason, in Sec. 8.2.2, we compress the belief of A in M, summarizing it with fewer representative points.

Reciprocal Sampling

To accelerate convergence, Prorok et al. [114], during the particle filter resampling step, propose sampling from the detection distribution (i.e., the right-most factor in Eq. (8.1)) 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. 8.2.

8.2.2 Distribution Compression

We present our approach for near-linear time distribution compression, based on Compress++ [136]. 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 [31]. An important measure is the maximum mean discrepancy (MMD) [44], 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

$$\mathrm{MMD}(X_1, X_2) \doteq ||\mathbb{E}_{X_1}[K(X_1)] - \mathbb{E}_{X_2}[K(X_2)]||_{\mathcal{H}}, \tag{8.4}$$

where $k(\cdot)$ is the reproducing kernel, such that $K(X) \doteq \left(k(x_i, x_j)\right)_{ij}$ is a symmetric positive semi-definite matrix over all input points $x_i \in X$. Kernel thinning (KT) algorithms [30, 31] 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. [136] 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

$$\bar{X}_t^{AB} = \{T(d_t; \mathbf{x}^A) | (\mathbf{x}^A, \cdot) \in X_t^A\} \subset \mathbb{R}^2 \tag{8.5}$$

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. (8.2) and Eq. (8.3) 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. (8.3):

$$\mu \sim \mathcal{U}(\tilde{X}_t^{AB}), \quad (x, y)_{t,i}^B \sim \mathcal{N}(\mu, \Sigma), \quad \theta_{t,i}^B \sim \mathcal{U}([0, 2\pi)). \tag{8.6}$$

8.2.3 Baselines

As part of our contribution, we implement several baselines, including seminal works [40, 114] in collaborative localization. We provide an overview of these approaches, using common notations for clarity.

Density Estimation Tree

Fox et al. [40] propose to use density estimation trees (DET) [105, 118] 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:

$$p\left(\mathbf{x}_{t,i}^B \middle| M\left(d_t, X_t^A\right)\right) = \mathrm{DET}^{AB}((x, y)_{t,i}^B). \tag{8.7}$$

While DET was suggested as a remedy for the nontrivial issues of establishing correspondence between two sample sets (X^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 work does not include it.

Divide-and-Conquer Clustering

Prorok et al. [114] propose a non-iterative, order-independent, non-parametric clustering inspired by multidimensional binary trees [8]. The particles of A are clustered into K cluster abstractions, which include the centroid $mathbf{c}^A_{t,k}$, weight w^A_k, detection mean μ^A_k and detection variance Σ^A_k. The belief is updated as

$$p\left(mathbf{x}^B_{t,i}\middle|M\left(d_t, X^A_t\right)\right) = \sum_{k=1}^{K} w^A_k \mathcal{N}(T^{-1}\left((x,y)^B_{t,i}; \mathbf{c}^A_{t,k}\right); \mu^A_k, \Sigma^A_k + \Sigma),$$

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

K-means Clustering

K-means clustering [50] is a commonly-used, low-cost iterative clustering method. It is mentioned by Prorok et al. [114] 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 $\bar{X}^{AB}_t$ with centroids $\mathbf{c}^A_{t,k}$; for each cluster, we compute its intra-cluster variance Σ^A_k and total weight w^A_k. The belief is then updated as

$$p\left(\mathbf{x}^B_{t,i}\middle|M\left(d_t, X^A_t\right)\right) = \sum_{k=1}^{K} w^A_k \mathcal{N}\left((x,y)^B_{t,i}; \mathbf{c}^A_{t,j}, \Sigma^A_k + \Sigma\right) \tag{8.8}$$

By clustering $\bar{X}^{AB}_t$ instead of X^A, we reduce the amount of information to broadcast compared to the approach suggested by Prorok et al. [114]. For the reciprocal sampling strategy, we sample from the mixture of normal distributions of Eq. (8.8).

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. (8.2). In our implementation, we also include a reciprocal sampling step similar to the one preformed for our Compress++ approach.

8.3 Complexity Analysis

The complexity of all approaches is presented in Tab. 8.1. 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.

8.3.1 Compression

The naive implementation for belief exchange require just $O(N)$ complexity for compression, but results in high communication cost and computational cost during fusion. 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. [114], the complexity for K clusters is $O(NK)$. The complexity of K-means

Table 8.1. 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	$O(N^2)$	$O(NK)$	$O(N \log N)$	$O(NK)$	$O(NK)$	$O(N\sqrt{N})$

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. 8.5.3). 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 $O(N \log^3 N)$.

8.3.2 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 ([40]), 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).

8.3.3 Fusion

Following Eq. (8.2), the complexity of updating the belief is $O(N^2)$ for the naive implementation. Fox et al. [40] requires $O(N \log N)$, since it involves querying a tree. The update step, combined with the reciprocal sampling, detailed by Prorok and Martinoli [112] 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).

8.4 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.

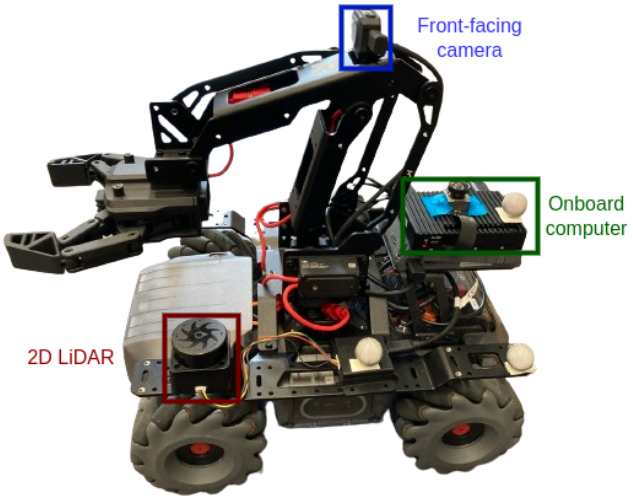


Figure 8.3. The robotic platform used in the evaluation.

8.4.1 Robots

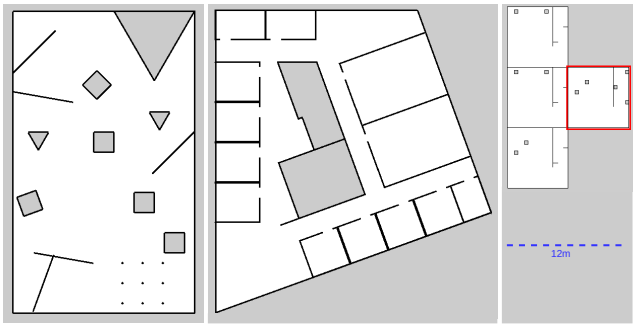
The platform we use in the evaluation is the DJI RoboMaster EP, a commercially available ground robot, with omnidirectional drive, and a size of 32 cm × 24 cm × 27 cm. Depicted in Fig. 8.3, the robot has a front-facing CMOS HD camera with a 102° horizontal FoV and a YDLIDAR Tmini Pro 2D LiDAR with a range of 12 m, 360° FoV, and resolution of 0.54° at 6 Hz. The onboard robot firmware includes an ML model for detecting other RoboMasters in the camera stream. The detector runs at about 5 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 $d = (r, \theta)$, with a zero-mean Gaussian error ($\sigma_r \approx 0.05r, \sigma_\theta \approx 0.03$ rad) that depends on the range. The robots carry a Single Computer Board (Khadas VIM4) that runs ROS2 drivers for the robot platform[1] and LiDAR.

The same platform is available in simulation[2] (CoppeliaSim [121]), where we replicate the same sensing error model and run the same ROS2 driver.

8.4.2 Environments

In simulation, we design three different environments (Fig. 8.4), 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 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 m^2 where experiments are executed with 10000 particles. Environment 3 is smaller, with a free area of 140 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.

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



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

Figure 8.4. 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.

Table 8.2. Common MCL parameters

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

8.4.3 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. 8.2.

8.4.4 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 m radius of the ground truth pose, and the orientation is within 0.3 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.

8.4.5 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. 8.2 on the same runs, using the parameters reported in Tab. 8.2 for the common MCL part. Overall, 112 evaluations were run for each of the 7 methods. The Compress++ algorithm reduces the

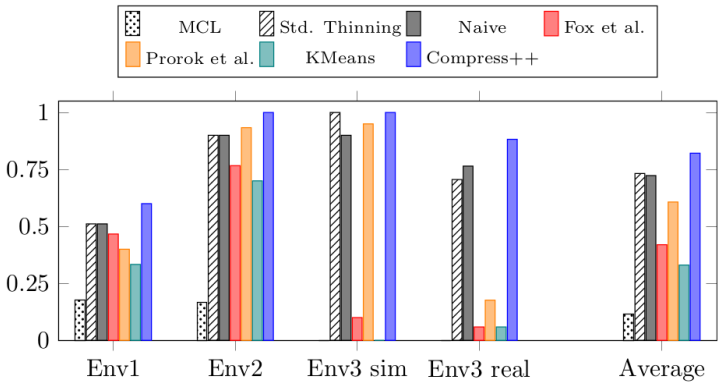


Figure 8.5. The success rate of all methods for each of the environments for robot B.

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 and Martinoli [112] explored different cluster numbers, between 1 and 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 and Martinoli [112].

8.5 Experimental Evaluation

We conducted a thorough evaluation of the different approaches to cooperative localization. We present our experiments to show the capabilities of our method. The results 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.

8.5.1 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. 8.5 and Tab. 8.3, the results highlight the importance of reciprocal sampling. The seminal work by Fox et al. [40] 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. 8.2, 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. (8.1) 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.

Table 8.3. 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

Another interesting insight comes from the difference in performance in different environments. Due to the sparsity of environment 1, 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. 8.3) 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. 8.6, 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. 8.5 and Tab. 8.3 success requires that pose estimation remains within a convergence threshold from the moment of convergence, in Fig. 8.6, we only consider whether a pose estimation is close enough to the ground truth at a given time. Therefore, Fig. 8.6 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.

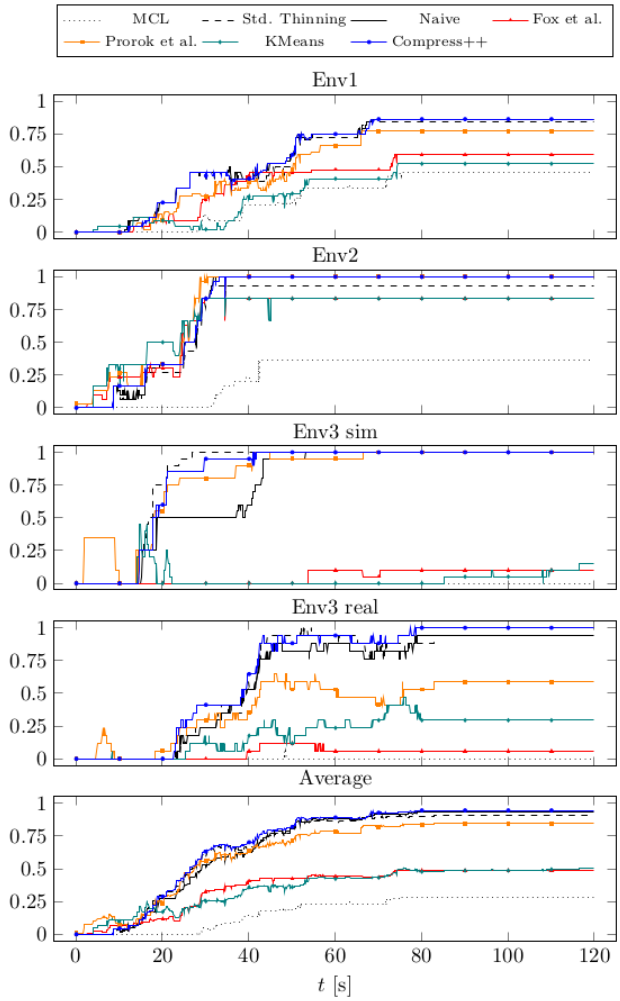


Figure 8.6. The fraction of runs whose current pose estimation of robot B is below convergence threshold. The time $t = 0$ is synchronized by the arrival of the first detection message from robot A.

Table 8.4. 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

8.5.2 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 $T = 20$. 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.

8.5.3 Runtime cost

As discussed in Sec. 8.3, 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. 8.4. 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.

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 a open-source Python implementation of Compress++.[3] Nonetheless, its runtime for compression is on-par with the approach of

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

(a) (b)

Figure 8.7. The behavior of different distribution compression methods on different data points formations.

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.

8.5.4 Clustering

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. 8.7a) and another 20 points from a region at the center (Fig. 8.7b). We then apply different methods to extracts 8 representative points.

This test case reveals a weakness of the divide-and-conquer clustering algorithm proposed by Prorok et al. [114]: as shown in Fig. 8.7b, 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.

8.6 Conclusion

In this chapter, 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.

Chapter 9

Conclusions

The objective of this dissertation was to address the challenges of robust long-term localization in changing, human-oriented environments using floor plans. The research we conducted to reach our objective included: (i) exploring additional sources of information to improve on the conventional geometry-based localization; (ii) exploiting semantic information for the task of global localization, inspired by human navigation; (iii) proposing fusion strategies for integrating information coming from different sensors or perception models; (iv) utilizing floor plans as a powerful prior for both long-term localization and mapping; (v) verifying that our research is applicable for a variety of robotic platform, including resource-constrained platforms.

The concept of enriching the traditional geometry-based MCL framework with additional sources of information is shared throughout the entire research process. In Sec. 4, we demonstrated the benefits of integrating textual cues from cameras into a range-based MCL. In Sec. 5, we showed how semantic information from objects can further improve the robustness of long-term localization, and how hierarchical view of semantic information can accelerate convergence. In Sec. 8, we illustrated the contribution of information exchange between robots to improve global localization.

The understanding that semantic information is powerful and should be leveraged for robot localization has guided this research from beginning to end. The benefits of semantic information are especially highlighted in Sec. 5 and Sec. 6, where semantic cues are used for localization and also for mapping. The strength of semantic cues is particularly evident when localization is greatly improved even when the semantic information is abstract and imprecise. Unlike geometry-based approaches that require great detail to enable global localization, utilizing semantic cues is advantageous even when objects are merely described with manually-annotated bounding box. This allows abstract semantic maps to be annotated and updated by an non-expert without a cumbersome mapping procedure.

A major factor in the robustness of localization is integration of multiple sensors and perception models. The issue of noisy, limited sensors or imprecise perception models can be mitigated by fusing information from multiple sources instead of relying solely on one. We filtered LiDAR measurements of dynamic objects using camera-based object detection in Sec. 5, and proposed an approach that factors in information from both camera and LiDARs. We combined LiDAR odometry with floor plan prior to assist with our camera-based semantic mapping in Sec. 6. We also showcased the ability of 3D object detection as a source of geometric information that can complement range sensors in localization. We proposed a sensor model for fusing short-sighted

miniaturized ToF sensors with cameras in Sec. 7. In Sec. 8 we integrate the camera-based perception model output of another robot to improve the range-based localization carried out by another robot.

As we were targeting indoor localization, and floor plans are generally available for human-oriented spaces such as offices and hospitals, we adopted floor plans as a prior since beginning. In all of our works, we proved that floor plans, despite being a sparse and incomplete representation of the scene's geometry, still provide valuable information and are easily augmented by semantic cues.

Online and onboard execution of our proposed approaches had been an essential and constant part of our evaluation. As the nature of robotics is practical, testing our works on different types of robots with a range of sensing capabilities and computational resources, was an integral part of our research. We have successfully deployed our approaches on medium and small wheeled-robots, as well as nano-drones, confirming that the principles that guided our research can be applied to robots in general.

Research on indoor localization has been particularly challenging and often non-conclusive due to several factors. For outdoor localization, there are variety of public, sensor-rich datasets which include ground-truth poses [42, 62, 100], some even include semantic annotations [6, 100]. However, for indoor localization there are no public datasets for indoor localization which contain multiple sensors and reliable, continuous ground truth poses. During my time in the university of Bonn, we invested in installing the infrastructure and designing the robots that would serve to create such dataset. However, after my departure, these plans had been mostly abandoned.

Another issue is the lack of open-source code for indoor localization papers. To truly determine the progress we have made with this dissertation, we would ideally compare ourselves to the state-of-the-art localization approaches that preceded us. Unfortunately, the relevant works did not include code. In cases where the code was straight-forward enough [39, 40, 113], we have taken upon ourselves to implement the baselines, and open-sourced the code as a contribution to our community. However, papers that are not as clear or make use of a fine-tuned neural network model cannot be re-implemented. We see the lack of open-source code as a major detriment to the progress of indoor localization, and therefore made the implementations to all our work publicly available.

Last but not least, is the absence of standard metrics to evaluate the quality of indoor localization. A variety of metrics are proposed in the relevant literature, and even when the same metrics (in principle) are used, their implementation may differ and result in substantially different values. Domains that experienced rapid progress in the last decade, are also known to have prevalent usage of standard metrics, such as average-precision and intersection-over-union which evaluate object detection and semantic segmentation. A famous benchmark for outdoor odometry is the KITTI benchmark [42] which popularized and standardize the implementation of metrics for evaluating odometry. These metrics, as well as the ones used in the related research for indoor localization and our own work, focus on geometric accuracy and evaluate localization in isolation. However, the metrics should reflect how much we contributed to enabling robot autonomy, which has been our main motivation all along. In the pursuit of questionable benchmarks, the focus on geometry-based approaches overshadowed research about the hierarchical, semantic description of the environment.

The abundance of research on sub-centimeter accurate localization and precise 3D reconstruction of large scenes begs the question, did we forget our goal as roboticists? Localization and mapping transformed into a competitive sport, without considering the fact that these two

tasks are in service of a higher purpose - enabling more complex abilities and behaviors that will allow robots to assist humans. From this perspective, of being a small (but crucial) part in a larger system, the mindset behind localization and mapping approaches and how we evaluate them, need to shift drastically. And this, I hope to explore in my future research.

Appendix A

Publications

First authors (including equal contributors) are marked in bold.

A.1 Long-term Localization

We presented Long-term localization exploiting textual information in [170]. We extended the work to robust localization using semantic cues in user annotated abstract semantic maps in [171]. We proposed an approach for 3D metric-semantic mapping for long-term localization in [172].

[170] **Zimmerman, Nicky** and Wiesmann, Louis and Guadagnino, Tiziano and LÃ¤be, Thomas and Behley, Jens and Stachniss, Cyrill. Robust Onboard Localization in Changing Environments Exploiting Text Spotting. *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2022.

[171] **Zimmerman, Nicky** and Guadagnino, Tiziano and Chen, Xieyuanli and Behley, Jens and Stachniss, Cyrill. Long-Term Localization Using Semantic Cues in Floor Plan Maps. *IEEE Robotics and Automation Letters (RA-L)*, 176–183, 2023.

[66] **Kuang, Haofei** and Chen, Xieyuanli and Guadagnino, Tiziano and Zimmerman, Nicky and Behley, Jens and Stachniss, Cyrill. IR-MCL: Implicit Representation-based Online Global Localization. *IEEE Robotics and Automation Letters (RA-L)*, 1627–1634, 2023.

[172] **Zimmerman, Nicky and Sodano, Matteo** and Marks, Elias and Behley, Jens and Stachniss, Cyrill. Constructing Metric-Semantic Maps Using Floor Plan Priors for Long-Term Indoor Localization. *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2023.

[158] **Wiesmann, Louis** and Guadagnino, Tiziano and Vizzo, Ignacio and Zimmerman, Nicky and Pan, Yue and Kuang, Haofei and Behley, Jens and Stachniss, Cyrill. LocNDF: Neural Distance Field Mapping for Robot Localization. *IEEE Robotics and Automation Letters (RA-L)*, 2023.

A.2 Resource-Constrained Localization

We addressed the challenges of localization under resource constraints in [97]. In [92] we proposed a lightweight object detection architecture that can be deployed to various MCUs, lay

the groundwork for our work on semantic localization under resource-constraints [174].

[97] **Müller, Hanna and Zimmerman, Nicky** and Polonelli, Tommaso and Magno, Michele and Behley, Jens and Stachniss, Cyrill and Benini, Luca. Fully On-board Low-Power Localization with Multizone Time-of-Flight Sensors on Nano-UAVs. *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2023. (Equal contribution)

[92] **Moosmann, Julian and Müller, Hanna and Zimmerman, Nicky** and Rutishauser, Georg and Benini, Luca and Magno, Michele. Flexible and Fully Quantized Ultra-Lightweight TinyissimoYOLO for Ultra-Low-Power Edge Systems.*IEEE Access*, 2024. (Equal contribution)

[174] **Zimmerman, Nicky and Müller, Hanna** and Magno, Michele and Benini, Luca. Fully Onboard Low-Power Localization with Semantic Sensor Fusion on a Nano-UAV using Floor Plans. *IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2024. (Equal contribution)

A.3 Collaborative Localization

[170] **Zimmerman, Nicky** and Giusti, Alessandro and Guzzi, Jérôme.Resource-Aware Collaborative Monte Carlo Localization with Distribution Compression. *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2024.

A.4 Human Pose Estimation on nano-UAV

In addition to the main research about localization, we also present some work on human pose estimation for nano-UAVs.

[108] **Palossi, Daniele** and Zimmerman, Nicky and Burrello, Alessio and Conti, Francesco and Müller, Hanna and Gambardella, Luca Maria and Benini, Luca and Giusti, Alessandro and Guzzi, Jérôme. Fully onboard ai-powered human-drone pose estimation on ultralow-power autonomous flying nano-uavs. *IEEE Internet of Things Journal*, 1913–1929, 2023.

[20] **Cereda, Elia** and Ferri, Marco and Mantegazza, Dario and Zimmerman, Nicky and Gambardella, Luca M. and Guzzi, Jérôme and Giusti, Alessandro and Palossi, Daniele. Improving the Generalization Capability of DNNs for Ultra-low Power Autonomous Nano-UAVs. *17th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 327–334, 2021.

A.5 Software Releases

Open-source code drives the robotics community forward, and therefore, we strive to release the implementation for our publications. Our papers [170, 171, 97, 172, 66, 158, 174, 173] have publicly available code on Github.

Bibliography

[1] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-Squares Fitting of Two 3-D Point Sets. *IEEE Trans. on Pattern Analalysis and Machine Intelligence (TPAMI)*, PAMI-9(5):698–700, 1987.

[2] 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, 2016. doi: 10.1177/0278364915596589.

[3] R. Azzam, T. Taha, S. Huang, and Y. Zweiri. Feature-based visual simultaneous localization and mapping: A survey. *SN Applied Sciences*, 2:1–24, 2020.

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

[5] L. E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state Markov chains. *The annals of mathematical statistics*, 37(6):1554–1563, 1966.

[6] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2019.

[7] M. Bennewitz, C. Stachniss, W. Burgard, and S. Behnke. Metric Localization with Scale-Invariant Visual Features using a Single Perspective Camera. In H. Christiensen, editor, *European Robotics Symposium 2006*, volume 22 of *STAR Springer Tracts in Advanced Robotics*, pages 143–157. Springer Verlag, 2006. ISBN 3-540-32688-X. URL `http://www.informatik.uni-freiburg.de/~stachnis/pdf/bennewitz06euros.pdf`.

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

[9] M. Betke and L. Gurvits. Mobile robot localization using landmarks. *IEEE transactions on robotics and automation*, 13(2):251–263, 1997.

[10] P. Biber and T. Duckett. Dynamic Maps for Long-Term Operation of Mobile Service Robots. In *Proc. of Robotics: Science and Systems (RSS)*, pages 17–24, 2005.

[11] H. Blum, J. Stiefel, C. Cadena, R. Siegwart, and A. Gawel. Precise robot localization in architectural 3d plans. *arXiv preprint*, 2020.

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

[13] F. Boniardi, T. Caselitz, R. Kümmerle, and W. Burgard. Robust LiDAR-based localization in architectural floor plans. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2017.

[14] F. Boniardi, A. Valada, R. Mohan, T. Caselitz, and W. Burgard. Robot localization in floor plans using a room layout edge extraction network. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[15] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[16] G. Brazil, J. Straub, N. Ravi, J. Johnson, and G. Gkioxari. Omni3D: A Large Benchmark and Model for 3D Object Detection in the Wild. *arXiv preprint:2207.10660*, 2022.

[17] R. A. Brualdi and V. S. Pless. Greedy Codes. *Journal of Combinatorial Theory, Series A*, 64(1):10–30, 1993. ISSN 0097-3165. doi: https://doi.org/10.1016/0097-3165(93)90085-M. URL https://www.sciencedirect.com/science/article/pii/009731659390085M.

[18] 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. URL http://arxiv.org/pdf/1606.05830v4.

[19] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 213–229. Springer, 2020.

[20] E. Cereda, M. Ferri, D. Mantegazza, N. Zimmerman, L. M. Gambardella, J. Guzzi, A. Giusti, and D. Palossi. Improving the Generalization Capability of DNNs for Ultra-low Power Autonomous Nano-UAVs. In *2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 327–334, 2021. doi: 10.1109/DCOSS52077.2021.00060.

[21] N. Chebrolu, T. Läbe, O. Vysotska, J. Behley, and C. Stachniss. Adaptive Robust Kernels for Non-Linear Least Squares Problems. *IEEE Robotics and Automation Letters (RA-L)*, 6 (2):2240–2247, 2021.

[22] X. Chen, A. Milioto, E. Palazzolo, P. Giguére, J. Behley, and C. Stachniss. SuMa++: Efficient LiDAR-based Semantic SLAM. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019. URL http://www.ipb.uni-bonn.de/wp-content/papercite-data/pdf/chen2019iros.pdf.

[23] 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 Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. URL https://openaccess.thecvf.com/content_CVPR_2020/papers/Cheng_Panoptic-DeepLab_A_Simple_Strong_and_Fast_Baseline_for_Bottom-Up_Panoptic_CVPR_2020_paper.pdf.

[24] M. Coppola, K. N. McGuire, K. Y. Scheper, and G. C. de Croon. On-board communication-based relative localization for collision avoidance in micro air vehicle teams. *Autonomous Robots*, 42:1787–1805, 2018.

[25] B. Şimşek and H. Ş. Bilge. A Novel Motion Blur Resistant vSLAM Framework for Micro/Nano-UAVs. *Drones*, 5(4), 2021. ISSN 2504-446X. doi: 10.3390/drones5040121.

[26] 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.

[27] M. Cummins and P. Newman. FAB-MAP: Probabilistic localization and mapping in the space of appearance. *Intl. Journal of Robotics Research (IJRR)*, 27(6):647–665, 2008.

[28] 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.

[29] R. Douc and O. Cappe. Comparison of Resampling Schemes for Particle Filtering. In *ISPA 2005. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005.*, 2005.

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

[31] R. Dwivedi and L. Mackey. Generalized Kernel Thinning. In *International Conference on Learning Representations*, 2022.

[32] A. Elfes. *A probabilistic framework for robot perception and navigation*. PhD thesis, Carnegie-Mellon University, 1989.

[33] M. Elhousni and X. Huang. A Survey on 3D LiDAR Localization for Autonomous Vehicles. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, 2020.

[34] N. Elkunchwar, S. Chandrasekaran, V. Iyer, and S. B. Fuller. Toward battery-free flight: Duty cycled recharging of small drones. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5234–5241. IEEE, 2021.

[35] I. Fedorov, R. P. Adams, M. Mattina, and P. Whatmough. Sparse: Sparse architecture search for cnns on resource-constrained microcontrollers. *Advances in Neural Information Processing Systems*, 32, 2019.

[36] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Visual Object Detection with Deformable Part Models. *Commun. ACM*, 56:97–105, 2013.

[37] P. F. Felzenszwalb and D. P. Huttenlocher. Distance Transforms of Sampled Functions. *Theory of Computing*, 8(1):415–428, 2012.

[38] D. Fox. KLD-sampling: Adaptive particle filters. *Proc. of the Advances in Neural Information Processing Systems (NIPS)*, 14, 2001.

[39] 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.

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

[41] C. Friess, V. Niculescu, T. Polonelli, M. Magno, and L. Benini. Fully onboard slam for distributed mapping with a swarm of nano-drones. *arXiv preprint:2309.03678*, 2023.

[42] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[43] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014. URL `https://arxiv.org/pdf/1311.2524.pdf`.

[44] 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.

[45] M. Grinvald, F. Furrer, T. Novkovic, J. J. Chung, C. Cadena, R. Siegwart, and J. Nieto. Volumetric Instance-Aware Semantic Mapping and 3D Object Discovery. *IEEE Robotics and Automation Letters (RA-L)*, 4(3):3037–3044, 2019. ISSN 2377-3766.

[46] G. Grisetti, C. Stachniss, and W. Burgard. Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2005.

[47] J.-S. Gutmann. Markov-Kalman localization for mobile robots. In *2002 International Conference on Pattern Recognition*, 2002.

[48] N. Gyagenda, J. V. Hatilima, H. Roth, and V. Zhmud. A review of gnss-independent uav navigation techniques. *Robotics and Autonomous Systems*, page 104069, 2022.

[49] D. Hahnel, W. Burgard, D. Fox, K. Fishkin, and M. Philipose. Mapping and localization with RFID technology. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2004.

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

[51] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. URL `https://arxiv.org/pdf/1512.03385`.

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

[53] 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.

[54] 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.

[55] H. Howard-Jenkins and V. A. Prisacariu. LaLaLoc++: Global floor plan comprehension for layout localisation in unvisited environments. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2022.

[56] E. Impulse. FOMO: Real-Time Object Detection on Microcontrollers, a Presentation from Edge Impulse, 2022. URL https://www.edge-ai-vision.com/2022/06/fomo-real-time-object-detection-on-microcontrollers-a-presentation-from-edge-impulse/.

[57] 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. doi: 10.1109/ICRA.2014.6906890. URL http://ais.informatik.uni-freiburg.de/publications/papers/ito14icra.pdf.

[58] G. Jocher. ultralytics/yolov5: v3.1. https://github.com/ultralytics/yolov5, 2020. URL https://doi.org/10.5281/zenodo.4154370.

[59] G. Jocher, A. Chaurasia, and J. Qiu. YOLO by Ultralytics. https://github.com/ultralytics/ultralytics, Jan. 2023.

[60] 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.

[61] T. Ke and S. I. Roumeliotis. An efficient algebraic solution to the perspective-three-point problem. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[62] G. Kim, Y. S. Park, Y. Cho, J. Jeong, and A. Kim. MulRan: Multimodal Range Dataset for Urban Place Recognition. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2020.

[63] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár. Panoptic Segmentation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[64] I. Kostavelis and A. Gasteratos. Semantic mapping for mobile robotics tasks: A survey. *Journal on Robotics and Autonomous Systems (RAS)*, 66:86–103, 2015.

[65] T. Krajník, J. P. Fentanes, M. Hanheide, and T. Duckett. Persistent localization and life-long mapping in changing environments using the frequency map enhancement. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016.

[66] H. Kuang, X. Chen, T. Guadagnino, N. Zimmerman, J. Behley, and C. Stachniss. IR-MCL: Implicit representation-based online global localization. *IEEE Robotics and Automation Letters (RA-L)*, 8(3):1627–1634, 2023.

[67] H. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955. URL https://web.eecs.umich.edu/~pettie/matching/Kuhn-hungarian-assignment.pdf.

[68] J. Leonard and H. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Trans. on Robotics and Automation*, 7(3):376–382, 1991.

[69] V. Lepetit, F. Moreno-Noguer, and P. Fua. EP n P: An accurate O (n) solution to the P n P problem. *Intl. Journal of Computer Vision (IJCV)*, 81:155–166, 2009.

[70] K. Li, D. DeTone, Y. F. S. Chen, M. Vo, I. Reid, H. Rezatofighi, C. Sweeney, J. Straub, and R. Newcombe. ODAM: Object Detection, Association, and Mapping using Posed RGB Video. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2021.

[71] K. Li, H. Rezatofighi, and I. Reid. MOLTR: Multiple Object Localization, Tracking and Reconstruction From Monocular RGB Videos. *IEEE Robotics and Automation Letters (RA-L)*, 6(2):3341–3348, 2021.

[72] Z. Li, M. H. Ang, and D. Rus. Online Localization with Imprecise Floor Space Maps using Stochastic Gradient Descent. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[73] M. Liao, Z. Wan, C. Yao, K. Chen, and X. Bai. Real-time Scene Text Detection with Differentiable Binarization. *arXiv preprint*, 1911.08947, 2019.

[74] J. Lin, W.-M. Chen, Y. Lin, C. Gan, S. Han, et al. Mcunet: Tiny deep learning on iot devices. *Advances in Neural Information Processing Systems*, 33:11711–11722, 2020.

[75] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common Objects in Context. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 740–755, 2014.

[76] H. Liu, H. Darabi, P. Banerjee, and J. Liu. Survey of wireless indoor positioning techniques and systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(6):1067–1080, 2007.

[77] D. Lowe. Object recognition from local scale-invariant features. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, volume 2, pages 1150–1157, 1999. URL `http://www.cs.ubc.ca/~lowe/papers/iccv99.pdf`.

[78] R. Maffei, V. A. M. Jorge, V. F. Rey, M. Kolberg, and E. Prestes. Fast Monte Carlo Localization using spatial density information. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2015.

[79] R. Maffei, D. Pittol, M. Mantelli, E. Prestes, and M. Kolberg. Global localization over 2d floor plans with free-space density based on depth information. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[80] E. Marchand, H. Uchiyama, and F. Spindler. Pose estimation for augmented reality: a hands-on survey. *IEEE transactions on visualization and computer graphics*, 22(12):2633–2651, 2015.

[81] 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.

[82] J. McCormac, R. Clark, M. Bloesch, A. Davison, and S. Leutenegger. Fusion++: Volumetric Object-Level SLAM. In *Proc. of the Intl. Conf. on 3D Vision*, 2018.

[83] K. McGuire, C. De Wagter, K. Tuyls, H. Kappen, and G. C. de Croon. Minimal Navigation Solution for a Swarm of Tiny Flying Robots to Explore an Unknown Environment. *Science Robotics*, 4(35), 2019.

[84] O. Mendez, S. Hadfield, N. Pugeault, and R. Bowden. Sedar-semantic detection and ranging: Humans can localise without lidar, can robots? In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2018.

[85] O. Mendez, S. Hadfield, N. Pugeault, and R. Bowden. Sedar-semantic detection and ranging: Humans can localise without lidar, can robots? In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2018.

[86] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar. The GRASP Multiple Micro-UAV Testbed. *IEEE Robotics and Automation Magazine*, 17(3):56–65, 2010. doi: 10.1109/MRA.2010.937855.

[87] Microsoft. Neural Network Intelligence. `https://github.com/microsoft/nni`, 2021. URL `https://github.com/microsoft/nni`.

[88] M. Milford and G. Wyeth. SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2012.

[89] R. Miyagusuku, A. Yamashita, and H. Asama. Data information fusion from multiple access points for wifi-based self-localization. *IEEE Robotics and Automation Letters (RA-L)*, 4(2):269–276, 2018.

[90] R. Miyagusuku, A. Yamashita, and H. Asama. Data Information Fusion From Multiple Access Points for WiFi-Based Self-localization. *IEEE Robotics and Automation Letters (RA-L)*, 4(2):269–276, 2019.

[91] M. Moakher. Means and averaging in the group of rotations. *SIAM Journal on Matrix Analysis and Applications*, 24(1):1–16, 2002.

[92] J. Moosmann, H. Müller, N. Zimmerman, G. Rutishauser, L. Benini, and M. Magno. Flexible and Fully Quantized Lightweight TinyissimoYOLO for Ultra-Low-Power Edge Systems. *IEEE Access*, pages 1–1, 2024. doi: 10.1109/ACCESS.2024.3404878.

[93] H. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 1985.

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

[95] A. Mucherino, P. J. Papajorgji, and P. M. Pardalos. *k-Nearest Neighbor Classification*, pages 83–106. Springer Verlag, 2009.

[96] H. Müller, V. Niculescu, T. Polonelli, M. Magno, and L. Benini. Robust and Efficient Depth-based Obstacle Avoidance for Autonomous Miniaturized UAVs. *arXiv preprint*, 2022. doi: 10.48550/ARXIV.2208.12624. URL `https://arxiv.org/abs/2208.12624`.

[97] H. Müller, N. Zimmerman, T. Polonelli, M. Magno, J. Behley, C. Stachniss, and L. Benini. Fully On-board Low-Power Localization with Multizone Time-of-Flight Sensors on Nano-UAVs. In *Proc. of Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6. IEEE, 2023.

[98] 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.

[99] 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.

[100] T.-M. Nguyen, S. Yuan, T. H. Nguyen, P. Yin, H. Cao, L. Xie, M. Wozniak, P. Jensfelt, M. Thiel, J. Ziegenbein, and N. Blunder. MCD: Diverse Large-Scale Multi-Campus Dataset for Robot Perception. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[101] V. Niculescu, L. Lamberti, F. Conti, L. Benini, and D. Palossi. Improving Autonomous Nano-Drones Performance via Automated End-to-End Optimization and Deployment of DNNs. *IEEE Journal on Emerging and Selected Topics in Circuits and Sys.*, 11(4):548–562, 2021.

[102] V. Niculescu, D. Palossi, M. Magno, and L. Benini. Energy-efficient, Precise UWB-based 3-D Localization of Sensor Nodes with a Nano-UAV. *IEEE Internet of Things Journal*, 2022.

[103] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto. Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2017. URL `https://helenol.github.io/publications/iros_2017_voxblox.pdf`.

[104] E. Olson. AprilTag: A robust and flexible visual fiducial system. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2011.

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

[106] 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.

[107] E. Palazzolo, J. Behley, P. Lottes, P. Giguere, and C. Stachniss. ReFusion: 3D Reconstruction in Dynamic Environments for RGB-D Cameras Exploiting Residuals. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[108] D. Palossi, N. Zimmerman, A. Burrello, F. Conti, H. Müller, L. M. Gambardella, L. Benini, A. Giusti, and J. Guzzi. Fully onboard ai-powered human-drone pose estimation on ultralow-power autonomous flying nano-uavs. *IEEE Internet of Things Journal*, 9(3): 1913–1929, 2021.

[109] D. Palossi, N. Zimmerman, A. Burrello, F. Conti, H. Müller, L. M. Gambardella, L. Benini, A. Giusti, and J. Guzzi. Fully Onboard AI-Powered Human-Drone Pose Estimation on Ultralow-Power Autonomous Flying Nano-UAVs. *IEEE Internet of Things Journal*, 9(3): 1913–1929, 2022.

[110] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[111] P. Pfaff, W. Burgard, and D. Fox. Robust Monte-Carlo Localization Using Adaptive Likelihood Models. In *STAR Springer Tracts in Advanced Robotics*, 2006.

[112] 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.

[113] A. Prorok, A. Bahr, and A. Martinoli. Low-cost collaborative localization for large-scale multi-robot systems. In *2012 IEEE International Conference on Robotics and Automation*, pages 4236–4241. IEEE, 2012.

[114] 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.

[115] C. R. Qi, O. Litany, K. He, and L. J. Guibas. Deep Hough Voting for 3D Object Detection in Point Clouds. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2019.

[116] C. R. Qi, X. Chen, O. Litany, and L. J. Guibas. Imvotenet: Boosting 3d object detection in point clouds with image votes. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[117] N. Radwan, G. Tipaldi, L. Spinello, and W. Burgard. Do You See the Bakery? Leveraging Geo-Referenced Texts for Global Localization in Public Maps. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2016.

[118] P. Ram and A. G. Gray. Density estimation trees. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011.

[119] A. Ranganathan and F. Dellaert. Semantic modeling of places using objects. In *Proc. of Robotics: Science and Systems (RSS)*, 2007.

[120] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[121] 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.

[122] A. Rosinol, M. Abate, Y. Chang, and L. Carlone. Kimera: an open-source library for real-time metric-semantic localization and mapping. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2020.

[123] D. Rossi, F. Conti, M. Eggiman, A. D. Mauro, G. Tagliavini, S. Mach, M. Guermandi, A. Pullini, I. Loi, J. Chen, E. Flamand, and L. Benini. Vega: A Ten-Core SoC for IoT Endnodes With DNN Acceleration and Cognitive Wake-Up From MRAM-Based State-Retentive Sleep Mode. *IEEE Journal of Solid-State Circuits*, 57(1):127–139, 2022. doi: 10.1109/JSSC.2021.3114881.

[124] A. Rottmann, O. Martínez-Mozos, C. Stachniss, and W. Burgard. Place Classification of Indoor Environments with Mobile Robots using Boosting. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, pages 1306–1311, Pittsburgh, PA, USA, 2005. URL `http://www.informatik.uni-freiburg.de/~stachnis/pdf/rottmann05aaai.pdf`.

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

[126] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[127] M. Runz, M. Buffier, and L. Agapito. MaskFusion: Real-Time Recognition, Tracking and Reconstruction of Multiple Moving Objects. In *Proc. of the Intl. Symposium on Mixed and Augmented Reality (ISMAR)*, 2018.

[128] M. Runz, K. Li, M. Tang, L. Ma, C. Kong, T. Schmidt, I. Reid, L. Agapito, J. Straub, S. Lovegrove, and R. Newcombe. FroDO: From Detections to 3D Objects. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. URL `proceedings: runz2020cvpr.pdf`.

[129] J. Saarinen, H. Andreasson, T. Stoyanov, J. Ala-Luhtala, and A. J. Lilienthal. Normal distributions transform occupancy maps: Application to large-scale online 3D mapping. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2013.

[130] J. Schneider and W. Förstner. Real-time Accurate Geo-localization of a MAV with Omnidirectional Visual Odometry and GPS. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2014.

[131] J. Schneider, F. Schindler, T. Läebe, and W. Föerstner. Bundle Adjustment for Multi-camera Systems with Points at Infinity. In *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2012.

[132] J. Schneider, C. Eling, L. Klingbeil, H. Kuhlmann, W. Förstner, and C. Stachniss. Fast and Effective Online Pose Estimation and Mapping for UAVs. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2016.

[133] A. Segal, D. Haehnel, and S. Thrun. Generalized-icp. In *Proc. of Robotics: Science and Systems (RSS)*, 2009.

[134] D. Seichter, M. Köhler, B. Lewandowski, T. Wengefeld, and H.-M. Gross. Efficient RGB-D Semantic Segmentation for Indoor Scene Analysis. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021.

[135] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, and M. Guizani. Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges. *Ieee Access*, 7:48572–48634, 2019.

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

[137] B. Shi, X. Bai, and C. Yao. An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition. *arXiv preprint*, 1507.05717, 2015.

[138] S. Shi, X. Wang, and H. Li. PointRCNN: 3D Object Proposal Generation and Detection From Point Cloud. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[139] C. C. Slama, C. Theurer, and S. W. Henriksen. *Manual of photogrammetry*. American Society of Photogrammetry, Falls Church, 4th edition, 1980.

[140] 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.

[141] C. Stachniss and W. Burgard. Mobile Robot Mapping and Localization in Non-Static Environments. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, pages 1324–1329, 2005. URL `http://www.informatik.uni-freiburg.de/~stachnis/pdf/stachniss05aaai.pdf`.

[142] J. StÃ¼ckler, N. Biresev, and S. Behnke. Semantic mapping using object-class segmentation of RGB-D images. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2012. doi: 10.1109/IROS.2012.6385983.

[143] D. Sun, F. Geißer, and B. Nebel. Towards effective localization in dynamic environments. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016.

[144] N. SÃ¼nderhauf, F. Dayoub, S. McMahon, B. Talbot, R. Schulz, P. Corke, G. Wyeth, B. Upcroft, and M. Milford. Place categorization and semantic mapping on a mobile robot. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2016.

[145] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust Monte Carlo Localization for Mobile Robots. *Artificial Intelligence*, 128(1-2), 2001. URL `https://www.sciencedirect.com/science/article/pii/S0004370201000698`.

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

[147] G. D. Tipaldi, D. Meyer-Delius, and W. Burgard. Lifelong localization in changing environments. *Intl. Journal of Robotics Research (IJRR)*, 32(14):1662–1678, 2013.

[148] R. Valencia, J. Saarinen, H. Andreasson, J. Vallvé, J. Andrade-Cetto, and A. J. Lilienthal. Localization in highly dynamic environments using dual-timescale NDT-MCL. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2014.

[149] S. Van Der Helm, M. Coppola, K. N. McGuire, and G. C. de Croon. On-Board Range-Based Relative Localization for Micro Air Vehicles in Indoor Leader–Follower Flight. *Autonomous Robots*, 44(3):415–441, 2020.

[150] O. Vysotska and C. Stachniss. Exploiting Building Information from Publicly Available Maps in Graph-Based SLAM. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016. URL `http://www.ipb.uni-bonn.de/pdfs/vysotska16iros.pdf`.

[151] O. Vysotska and C. Stachniss. Lazy Data Association For Image Sequences Matching Under Substantial Appearance Changes. *IEEE Robotics and Automation Letters (RA-L)*, 1 (1):213–220, 2016. URL `http://www.ipb.uni-bonn.de/pdfs/vysotska16ral-icra.pdf`.

[152] K. Wada, E. Sucar, S. James, D. Lenton, and A. J. Davison. MoreFusion: Multi-object Reasoning for 6D Pose Estimation from Volumetric Fusion. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[153] T. Wang, X. Zhu, J. Pang, and D. Lin. FCOS3D: Fully convolutional one-stage monocular 3d object detection. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2021.

[154] X. Wang, R. J. Marcotte, and E. Olson. GLFP: Global Localization from a Floor Plan. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[155] Y. Wang, V. C. Guizilini, T. Zhang, Y. Wang, H. Zhao, and J. Solomon. DETR3D: 3D object detection from multi-view images via 3D-to-2D queries. In *Proc. of the Conf. on Robot Learning (CoRL)*, 2022.

[156] P. Warden and D. Situnayake. *Tinyml: Machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers*. O'Reilly Media, 2019.

[157] Y. Watanabe, K. R. Amaro, B. Ilhan, T. Kinoshita, T. Bock, and G. Cheng. Robust Localization with Architectural Floor Plans and Depth Camera. In *2020 IEEE/SICE International Symposium on System Integration (SII)*, 2020.

[158] L. Wiesmann, T. Guadagnino, I. Vizzo, N. Zimmerman, Y. Pan, H. Kuang, J. Behley, and C. Stachniss. LocNDF: Neural Distance Field Mapping for Robot Localization. *IEEE Robotics and Automation Letters (RA-L)*, 2023.

[159] L. Wiesmann, T. Läbe, L. Nunes, J. Behley, and C. Stachniss. Joint Intrinsic and Extrinsic Calibration of Perception Systems Utilizing a Calibration Environment. *IEEE Robotics and Automation Letters (RA-L)*, 9(10):9103–9110, 2024. doi: 10.1109/LRA.2024.3457385.

[160] W. Winterhalter, F. Fleckenstein, B. Steder, L. Spinello, and W. Burgard. Accurate indoor localization for rgb-d smartphones and tablets given 2d floor plans. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2015.

[161] D. Wu and H. Su. An improved probabilistic approach for collaborative multi-robot localization. In *2008 IEEE International Conference on Robotics and Biomimetics*, pages 1868–1875. IEEE, 2009.

[162] X. Xin, J. Jiang, and Y. Zou. A review of Visual-Based Localization. In *Proceedings of the 2019 International Conference on Robotics, Intelligent Control and Artificial Intelligence*, 2019.

[163] B. Xu, W. Li, D. Tzoumanikas, M. Bloesch, A. Davison, and S. Leutenegger. MID-Fusion: Octree-based Object-Level Multi-Instance Dynamic SLAM. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2019.

[164] S. Xu, R. Chen, G. Guo, Z. Li, L. Qian, F. Ye, Z. Liu, and L. Huang. Bluetooth, floor-plan, and microelectromechanical systems-assisted wide-area audio indoor localization system: Apply to smartphones. *IEEE Transactions on Industrial Electronics*, 69(11):11744–11754, 2021.

[165] S. Yang and S. Scherer. Cubeslam: Monocular 3-D object SLAM. *IEEE Trans. on Robotics (TRO)*, 35(4):925–938, 2019.

[166] C. Yi, I. H. Suh, G. H. Lim, and B.-U. Choi. Bayesian robot localization using spatial object contexts. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2009.

[167] F. Zafari, A. Gkelias, and K. K. Leung. A Survey of Indoor Localization Systems and Technologies. *IEEE Communications Surveys Tutorials (CST)*, 21(3):2568–2599, 2019.

[168] H. Zhang, F. Li, S. Liu, L. Zhang, H. Su, J. Zhu, L. M. Ni, and H.-Y. Shum. DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection. *arXiv preprint*, 2203.03605, 2022.

[169] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning Deep Features for Scene Recognition using Places Database. In *Conference on Neural Information Processing Systems (NIPS)*, 2014.

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

[171] 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.

[172] 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.

[173] N. Zimmerman, A. Giusti, and J. Guzzi. Resource-Aware Collaborative Monte Carlo Localization with Distribution Compression, 2024.

[174] N. Zimmerman, H. Müller, M. Magno, and L. Benini. Fully Onboard Low-Power Localization with Semantic Sensor Fusion on a Nano-UAV using Floor Plans. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*. IEEE, 2024.