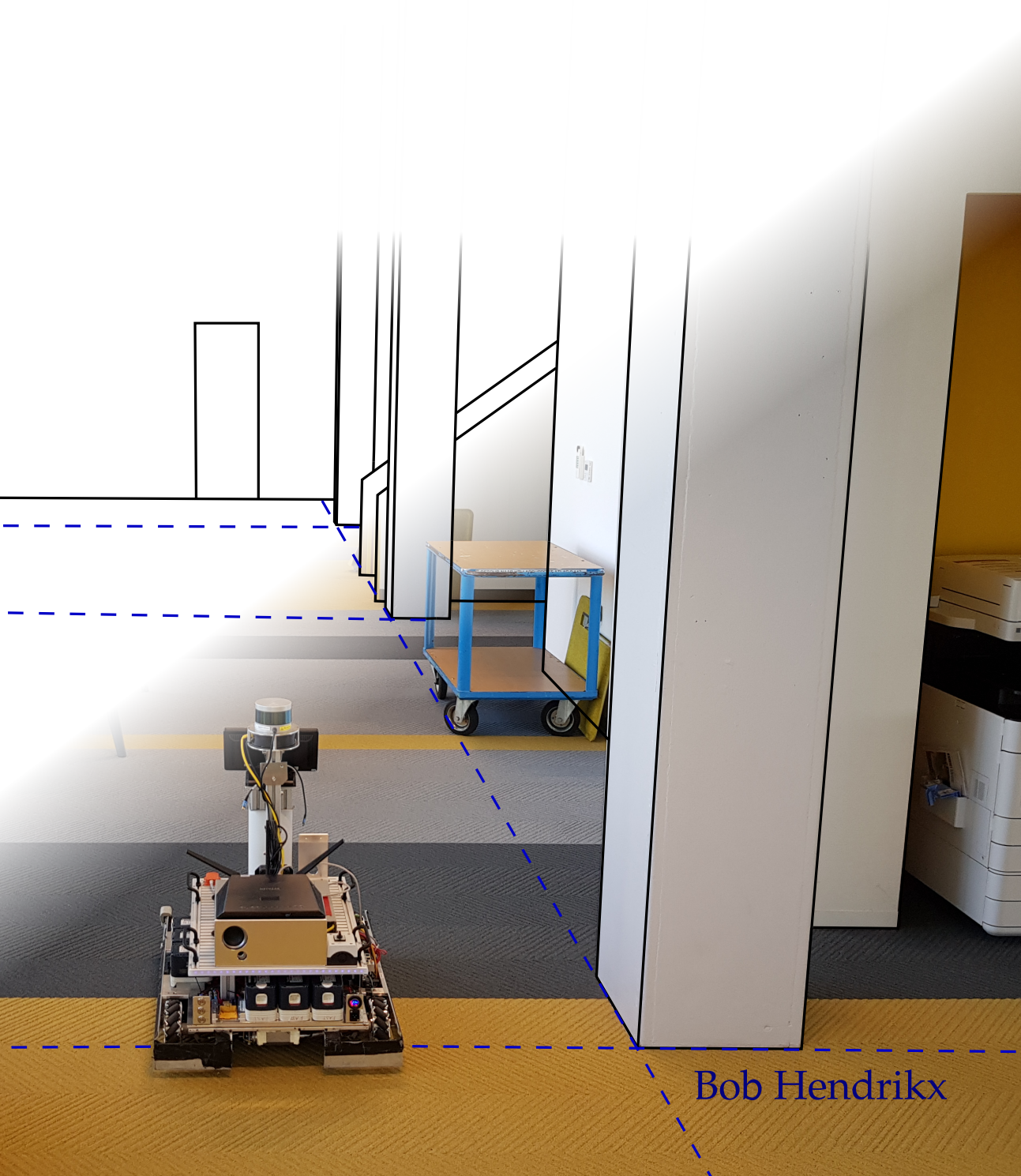
Object and Pattern Association for Robot Localization

Object and Pattern Association for Robot Localization



Bob Hendrikx

Object and Pattern Association for Robot Localization

R.W.M. Hendrikx

This work was carried out as part of the FAST research consortium.

A catalogue record is available from the Eindhoven University of Technology Library.
ISBN: 978-90-386-5719-6

Reproduction: www.proefschriftmaken.nl

Cover photo by Bob Hendrikx

Object and Pattern Association for Robot Localization

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
rector magnificus prof.dr.ir. F.P.T. Baaijens, voor een
commissie aangewezen door het College voor
Promoties, in het openbaar te verdedigen
op 12 April 2023 om 11:00 uur

door

Robertus Wilhelmus Maria Hendrikx

geboren te Wageningen

Dit proefschrift is goedgekeurd door de promotoren en de samenstelling van de promotiecommissie is als volgt:

voorzitter:	prof. dr. ir. P.D. Anderson
1e promotor:	prof. dr. ir. H.P.J. Bruyninckx
2e promotor:	dr. ir. M.J.G. van de Molengraft
copromotor:	dr. ir. J. Elfring
leden:	prof. dr. ir. M. Steinbuch
	dr. G. Dubbelman
	prof. dr. D. Nardi (Sapienza University of Rome)

Het onderzoek dat in dit proefschrift wordt beschreven is uitgevoerd in overeenstemming met de TU/e Gedragscode Wetenschapsbeoefening.

Summary

This thesis considers mobile robot localization, for which robots must be able to adequately detect and understand the world around them from onboard sensor data in order to relate that data to a map of the environment. In many cases, these maps do not represent the world exactly as it is perceived by the robot's sensors. If the localization strategy is not able to handle this mismatch by using hypotheses and being able to recover from false associations, the task will not succeed. This work considers indoor environments, for which only static elements and objects have been modeled on a map. The main objective of this thesis is to augment geometric maps with symbolic representations and to devise association-based algorithms that exploit these representations for robust localization, monitoring and recovery.

The contributions in this thesis consider a reference robot platform equipped with planar LiDAR and wheel encoder odometry. The first contribution is presented in Chapter 2, in which a method is proposed to use Building Information Models (BIM) from the built environment domain for robot localization. A graph-based model of the environment is created that connects the entities from a BIM model to geometric representations for the specific sensor. These representations are queried by the robot from a database and used for pose tracking using a factor graph. This graph contains explicit associations between the geometric features from the sensor and features on the map. The approach is experimentally demonstrated inside a university building for which a BIM model is available.

The second contribution is concerned with the question: When and how should a robot associate its sensor data with the map for global localization? A solution is proposed that splits the localization problem into two activities: 1) Maintaining a local map in the vicinity of the robot with features that can be reliably associated *locally* and 2) Evaluating the global associations whenever a new local stable feature enters the local map. The result is a feature association hierarchy in which explicit hypotheses are maintained to deal with ambiguous local features with respect to the global map. These hypotheses are evaluated and pruned by expanding a hypothesis tree and determining the lack-of-fit over a horizon of local features. The method is compared with a particle filter that relies on pose sampling and grid representations, and it is found that the proposed method successfully localizes in

more experimental runs.

In Chapter 4, this approach is extended with the exploitation of geometric patterns to better differentiate building elements from unmapped clutter. The association hierarchy is expanded by incorporating collinear and right-angled patterns that are represented as symbolic relations on the global map. These patterns occur naturally in many buildings and it is shown how they can help to discriminate between local observations and clutter observations. During localization, the patterns are extracted from a local map of individual features and are added to a hypothesis tree that is adapted to contain hypotheses on feature sets. It is shown how a basic set of spatial constraints can be used to find patterns and how relative feature distances within the pattern can be used to efficiently retrieve candidate patterns on the global map. An experiment with a fully autonomous robot is performed that prioritizes patterns in circular concrete columns over other structures. Furthermore, free space is used to monitor localization and to trigger recovery behavior that re-initializes the pattern search.

The approach to localization suggested in this thesis allows robots to perform robustly in cluttered environments by recognizing structure via explicit association with features and patterns. The result is a significant reduction in the amount of hypotheses needed. Furthermore, this explicit approach results in more self-explainable system behavior, which is a relevant property of an autonomous system that has to be accepted and understood in environments that are shared both with other systems and with humans.

Contents

Chapter 1

Introduction

1.1 Autonomous Mobile Robots

The field of robotics is concerned with designing and building systems that display a certain degree of autonomy in making decisions. These systems interact with the physical world and are therefore also known as *cyber-physical* systems, as they combine sensor inputs and actuators with software algorithms. In this thesis, the focus is on *mobile robotics*, which entails robot systems that are not fixed at some physical location, but can move around freely. Specifically, this thesis considers Autonomous Mobile Robots (AMRs), that use wheels to drive around in environments that have not been altered specifically for the robot. This sets AMRs apart from Autonomous Guided Vehicles (AGVs), which require special infrastructure such as sensor beacons, guiding strips or visual cues to be placed in the environment. In this section, the background, applications and challenges associated with this kind of system are introduced. A brief overview of the main parts of a mobile robot navigation stack is given together with relevant prior work and research questions, which are finally related to the contributions of this thesis.

1.1.1 Early days of mobile robotics

While mechanical systems that can run without intervention for periods of time have been envisioned and built for centuries, it was the invention of the computer (often attributed to Charles Babbage, see, e.g., [45]) that lead to the possibility of making these systems run *programs* that form the basis of sophisticated *robot behaviors*. One of the earliest conceptions of what would today be considered a *robot* (omitting science fiction works), is Shakey the robot [95]. What sets Shakey apart from other machines is its ability to plan for a sequence of actions that allows it to complete its task, whereas these actions would otherwise have to be supplied manually. The Shakey project lead to a number of important advances such as the

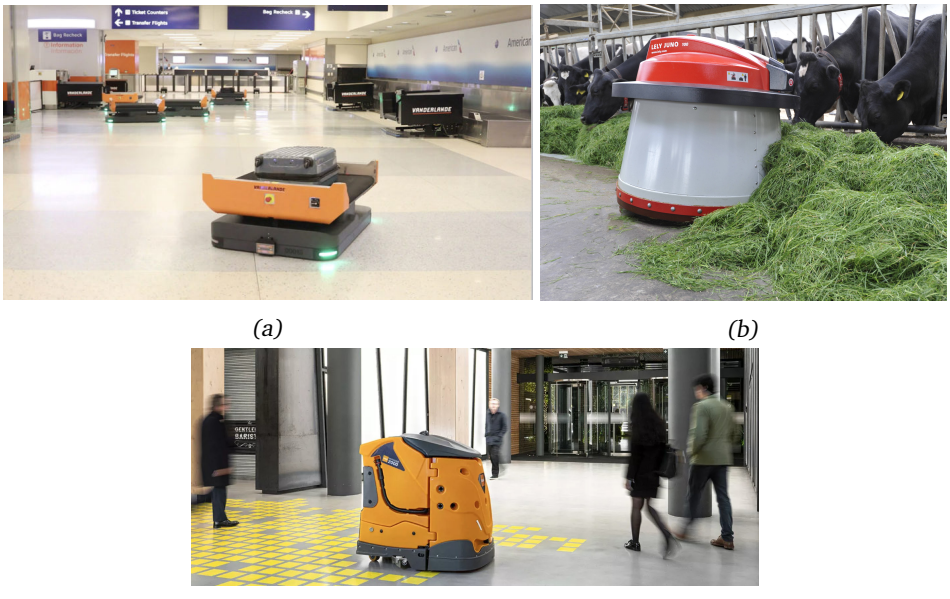
(a) *(b)* *(c)*

Figure 1.1. *(a) Shakey the robot. (b) An autonomous robot driving in crowded pedestrian zones [78]. (c) A service robot competing in the RoboCup tournament [130]*

A-star algorithm for planning and the Hough transform for detecting geometric features from sensor data, which are still widely used in robotics today. Figure 1.1 shows Shakey the robot, together with two more contemporary examples of research platforms. The platform in Figure 1.1b was used by researchers in Freiburg [78] to show robust outdoor navigation in dynamic city areas. It combines GPS with occupancy mapping for navigation. Figure 1.1c shows a Toyota Human Support Robot that is used for research in service robotics to assist people with daily tasks. For an overview of the history of mobile robots see, e.g., [23].

1.1.2 Real-world applications

The market for industrial mobile robots is projected to grow from USD 1.97 billion in 2021 to USD 8.70 billion by 2028 according to [5]. The current potential of wheeled robotics is mostly found in the ability to handle repetitive, dirty or dangerous tasks that otherwise would have to be performed by humans. Examples of domain where AMRs are already applied are transportation and warehouse automation [29], [26], inspection of large objects [2] or areas such as oil and gas sites [39] and cleaning automation. The latter has even emerged in consumer applications (e.g., iRobot [67]) next to the many existing industrial floor cleaning robots. Examples of modern industrial applications are shown in Figure 1.2; a robotic platform for luggage transportation and an automatic feed distributor inside a barn. Moreover, Figure 1.2c shows an example of a robot in a public space that is shared with humans. The multitude of applications has lead to a growing scientific interest in the area of *field robotics*, which is concerned with the challenges that robots have to face in the real world, as their tasks become more complex and their environments less predictable [80], [137]. Researched topics in this area range from perception and control to interaction and acceptance of humans that have to interact with mobile robots. The latter is an important part of many robot applications, as collaboration requires adequate interaction mechanisms [9] and



(a) (b)

(c)

Figure 1.2. *(a) A robot from Vanderlande moving luggage inside an airport [131]. (b) A feed-pushing robot from Lely operating on a farm [82]. (c) A cleaning robot from Taski, driving around in a public space [124].*

awareness of the personal space of people [3], [93]. Moreover, collaboration hinges on adequate *mental models* of the robots behavior because explainable behavior is an important prequisite for trust and effectiveness in human-robot collaboration [123].

1.1.3 Challenges and research questions

Robots have to know where they are in the environment and have to determine which actions to perform to achieve their navigation task. The focus of this thesis is robot localization: The robot associates the data from its sensors to objects represented on a map. That map is made beforehand, manually and/or based on previously processed sensor data. In both cases, a map can contain objects that might be relocated or are not present anymore, such as pieces of furniture. Similarly, the sensor data can come from objects that are not on the map. The following questions regarding localization motivate the work in this thesis:

(i) How can we interface mobile robots with existing building models from the built environment domain, such that these maps do not have to be created and maintained for a specific robot?

(ii) How can we make mobile robot localization robust against objects that are not represented on this map, but are visible in the robot's sensor data, if no prior estimate of location is available?

(iii) How can we add a layer of symbolic relations onto a map, that can help the robot differentiate between relevant observations and clutter.

In this thesis, the *semantic map* is considered as the central container for knowledge about the environment relating to localization and navigation. On this map, symbolic relations are added on top of a layer of geometric entities. Symbolic representations have several added values: (i) They allow computer programs to reference labels and relations from within relations, hence replacing to some extent "programming" by "modelling"; (ii) they improve algorithms that make data associations by providing information regarding data processing that is added a priori and (iii) they improve explainability and the ability to interact with other systems, operators and bystanders. They do so by condensing models and data into referable units. For example, the range readings from a Light Detection and Ranging (LiDAR) scan can be combined to form a line segment, which is hypothesized to be a wall, that is part of a building. This knowledge has implications for the hypothesized robot position with respect to the map, and possible actions that can be performed safely. In this thesis it is argued that the semantic map should not only be used to model the world, but also to configure the perception and motion of the robot via these declarative symbolic relations. This leads to increased *explainability*, by which we mean that a robot system can 1) provide insight into what it is doing and why it is doing it and 2) leverage this notion to reconfigure its own subsystems when necessary. These capabilities are still, however, far from realistic to expect from a modern robotic system [107], [18], and start with making representations more symbolic and hierarchical. This thesis takes a number of concrete steps in that direction for localization, taking the semantic map as a starting point.

1.2 Localization

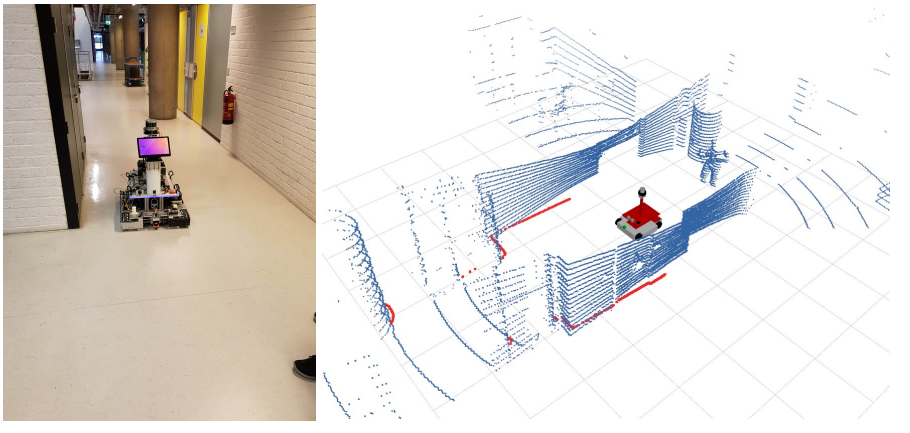
The focus of this thesis is on robot localization in indoor environments. The methods are demonstrated using planar LiDAR and wheel encoder odometry, but can be generalized to other sensor configurations. In general, requirements for successful localization depend on task and environment; while some tasks require high metric accuracy in a global reference frame, many tasks only require accuracy with respect to certain locally defined reference frame or a rough notion of correct heading. An example of the latter is a robot that navigates through a hallway for which it only needs to align its heading, until it passes through a narrow door by using motion control and local sensing. Moreover, accuracy does not only depend on the sensors and algorithms, but also on the global map representation. Therefore, a locally consistent view of the world based on sensor data is useful for both local motion control and global positioning, in which the latter is obtained by making

associations. The localization component of a mobile robot has to interact with components that perform planning and motion control of the platform. These components must interface with the sensor and actuators and these interfaces are described in hardware and software architectures. An overview of relevant concepts can be found in works such as [111], [18]. This section briefly introduces relevant concepts for robot navigation architectures, followed by a more specific overview of localization approaches and literature.

1.2.1 Hardware and software

The capabilities of mobile robots have increased significantly due to the sensors and processing algorithms that have become available in recent years. Firstly, sensors such as cameras and Light Detection And Ranging (LiDAR) devices have become widely available at low cost, incorporating large field-of-views with high accuracy and resolution. These sensors allow robots to accurately perceive the world around them, both in terms of geometry and in terms of attributes such as color. Secondly, computers have become increasingly powerful, enabling complicated *software stacks* consisting of multiple sensor processing and decision making algorithms to run concurrently on inexpensive embedded computers with small footprints and yet high communication bandwidths. Finally, information processing *paradigms* have been invented that allow robots to deal with uncertain information, originating from sensors and models of the world and of the outcome of actions that are performed. The probabilistic paradigm [127] is concerned with modeling this inherent uncertainty in prior knowledge and perception. Additionally, graph-based techniques have become popular for semantic world modeling and knowledge representation, and as a means to manage the complexity of probabilistic techniques [98]. The output of a planar LiDAR and a three-dimensional LiDAR are shown in Figure 1.3, together with the research platform that is used in these thesis and contains these sensors. It is evident that as sensors become more high-resolution, the amount of information becomes higher, requiring more processing power. One of the most popular software ecosystems in applied robotics that is used to manage information exchange between sensors and algorithms is the Robot Operating System (ROS) framework [106]. Besides providing a messaging system and an orchestration system for executable programs (ROS nodes), it also incorporates a package system with many (open source) implementations of popular sensor drivers, algorithms and map representations. The ROS ecosystem has played a significant role in making robotics research results readily available to anyone by providing relatively easy to use templates in the form of the ROS navigation stack. The applications in this thesis use the ROS middleware for communication with sensors and for visualization purposes.

There are many approaches to the design of a *navigation stack* or *navigation architecture* that closes the loop between sensing and control, as in Figure 1.4 (see, e.g. [17], [66]). An autonomous platform has to determine its actions based on its task, sensors and internal representation of the world. This is often achieved



(a) (b)

Figure 1.3. *(a) The robot platform used in this thesis, consisting of a holonomic (i.e., unidirectional) moving based and a separate sensor module mounted on top. (b) A visualization of sensor data from a planar LiDAR (Hokuyo utm-30lx) in red and from a 3D LiDAR in blue (Velodyne VLP-16). This thesis focuses on the planar sensor.*

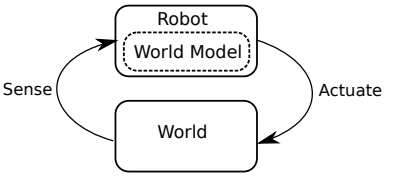


Figure 1.4. *The most basic conceptual model of a robot acting in an environment. The robot software contains an abstract world model, that is used to relate relevant information from sensor data to the state of the world. This state is then used to plan and perform actions that move the robot in the real world.*

by composing behaviors via a high-level coordination mechanism, such as a *state machine* or *behavior tree*. These behaviors (or states) are, for example, responsible for receiving task input, requesting a plan from a route planner and performing task-specific actions such as docking or other maneuvers. Such behaviors depend on perception and control components that often run as continuous processes that are interacted with via asynchronous communication. Many commonly used architectures heavily rely on planning based approaches for determining robot motion, whereas localization is often implemented as a continuous activity that provides the planning and execution modules with the robot pose in the map coordinate frame. An often made assumption here is that a reasonable initial estimate can be supplied (for example by the user or by a global positioning system), and keeping track of the robot location in the world model reduces to incremental *tracking*, using a proprioceptive sensor such as IMU or odometry and exteroceptive sensor to correct for the drift. The situation where no initial estimate is available is often referred to as *global localization*. Furthermore, an explicit connection between motion planning and localization is made in the field of *active localization* (see, e.g., [43], [110], [91], [76]). In active localization, robot actions are determined that take the uncertainty of the robot location into account. This thesis mainly focuses on global localization and heuristic policies that connect local sensor data to motion behaviors. Planning-based active localization is most relevant in environments where many locations appear highly similar in sensor data (such as environments with many similar looking connected hallways). In these cases, planning optimal motion actions to discriminate between locations within acceptable time is a relevant topic which is, however, outside the scope of this thesis.

1.2.2 Map representation

In this work, we define the *map* as a central instance-level representation of geometry and symbolic information about a particular indoor environment in which the robot is expected to perform its task. In general, this map can be a *prior* map or it can be built (over a chosen horizon) by the robot, using sensor data during *Simultaneous Localization and Mapping* (SLAM, [20]). Throughout this thesis the word *map* refers to a prior map, whereas we use the term *local* map for a map that is being constructed on the fly using only sensor data and for the robot's direct vicinity only. Local maps are constructed using SLAM algorithms and in this work, their main purpose will be to support global localization in the (prior) map.

Map representation plays an important role in the effectiveness and generality of localization methods. Four major classes of map representations are identified in literature: 1) sensor-data based representations, 2) feature representations, 3) cell decomposition representations and 4) object-based representations. Sensor data representations are popular for SLAM and localization and are often built from point clouds from LiDAR sensors (Figure 1.5e) or camera images that are stitched together into a consistent map [103]. Feature-based maps are a common representation

that contain primitive features either extracted from, or modeled for a specific sensor. Examples include visual features [112] or geometric features such as lines (Figure 1.5c) and planes. Such maps do not directly contain information about occupied and free space (which can be obtained if the pose of the imaging sensor is known for each raw sensor image). Cell decomposition methods do contain such information and are also very popular for indoor robotics, especially the grid map [37] representation and the voxel representation [69] (Figure 1.5a-b). These maps have the advantage that they can model both geometry as perceived by the sensor and traversable space for the robot. In their simplest form, such representations are relatively easy to generate from sensor data but do not model object instances. The object-based map is a very general and loosely defined concept that does model these instances, which can then be related to their geometric representations. These representations can consist of volumetric shapes or meshes, polygonal or curved geometries or collections of raw sensor data [105], [113], [31], [83], [92], [129]. In recent years, graph-based data models have received increasing attention as a mechanism for representing relations between geometric concepts and semantic information. An example is the addition of topological information about *places on the map*, the metric information associated with them and the connections between them that can be used for navigation [71], [8]. An example of this can be seen in Figure 1.5f where planes are extracted from LiDAR scans and are referenced in a semantic and topological layer.

For a comprehensive overview of graph-based modeling for the robotics domain, see [18]. In this thesis, we use both geometric models and graph models in order to describe objects and their sensor features. These representations are also popular in other domains, such as building representations [141], [140], [65].

1.2.3 Localization algorithms

Localization of a mobile robot requires a representation of the *state space* of robot locations. This state can be represented either continuous (i.e., 2D or 3D pose in a map frame) or discrete. Furthermore, topological representations represent localization by a combination of more loosely defined spaces and their connections, as opposed to a pose with respect to a map. Finally, association-based approaches handle localization via matches between sensor data and a map (or database) of reference images, scans or features. As these methods progress, their distinction may become less clear, as states are often used for determining likely associations and topological relations and vice versa.

A popular sample-based localization method that is often used for continuous state spaces is *particle filter* localization (also known as Monte Carlo localization), in which a set of hypotheses is maintained on the robot pose in the map [33], [89], [61], [6], [38]. The advantage of particle filters is that they can represent arbitrary distributions if enough samples are maintained and each sample can make a different data association between observations and the map. Handling these kinds of multi-modal distributions is essential for global localization. A

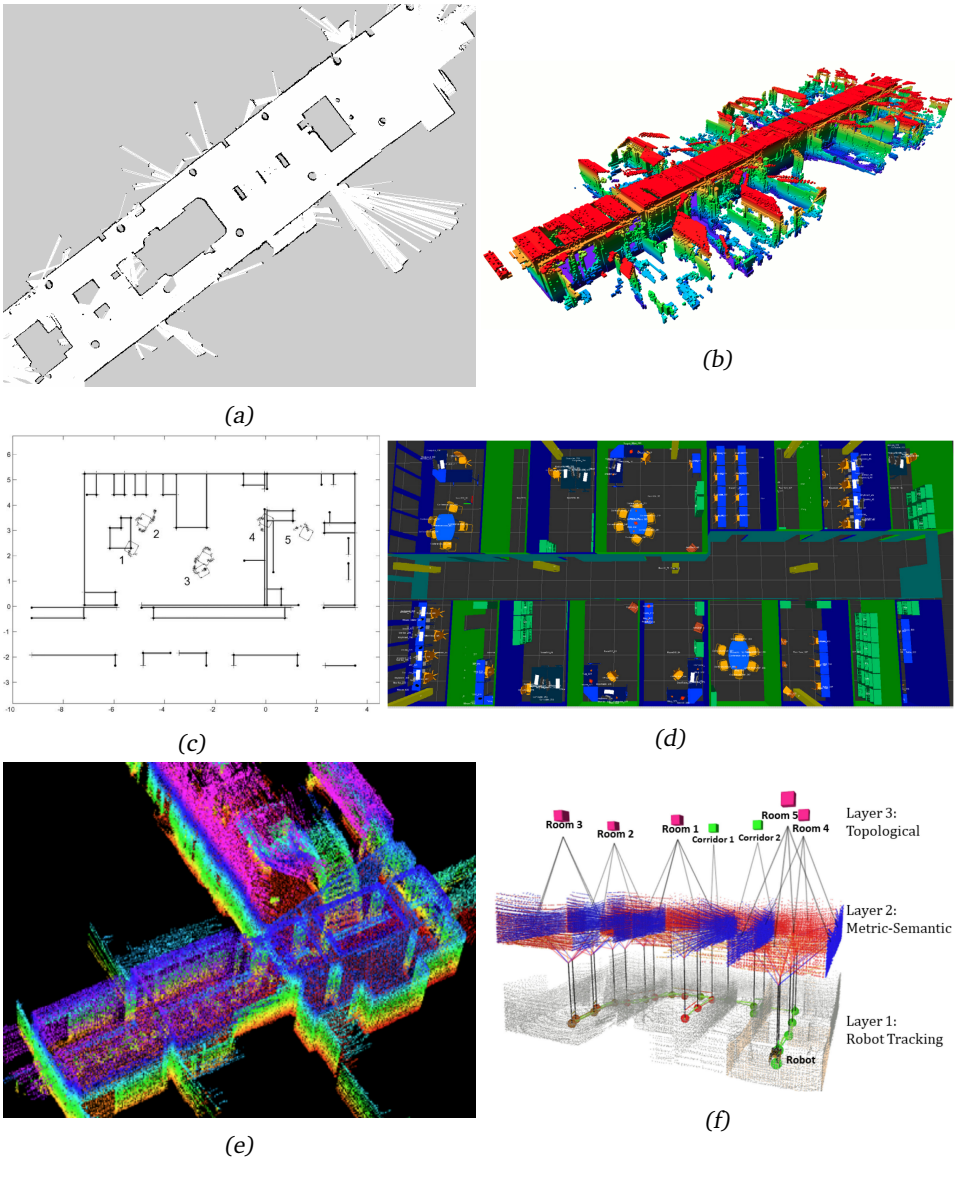


Figure 1.5. *(a) A gridmap representation of an environment that contains free and occupied cells (b) A voxel-based representation that contains three-dimensional occupancy information [69] (c) A vector-based representation consisting of line segments [4] (d) An object-based representation using volumetric shapes [31] (e) A pointcloud representation of a large lobby made by a SLAM algorithm [139] (f) A three-layered situational graph created by a SLAM algorithm [8]*

sensor model is used to describe the likelihood of the actual observation, given the expected observation for a particle. This sensor model is very general, and practical implementations for LiDARs can use beam-based models, likelihood field models, or feature based models. The latter requires the extraction of features from the LiDAR points, such as lines, corners, or other keypoint features. Particle filter methods can cope with significant initial uncertainty and global localization if the amount of particles is sufficient and the model of the environment is accurate enough. However, large particle sets are also associated with high computational cost and often contain many (approximately) *duplicate* hypotheses. They rely on the *Markov Assumption*, which means that all past sensor information is summarized in a current state estimate. Often, sensor readings result in highly symmetrical pose distributions (for example, a circle of possible locations around a point feature), for which sample-based representations are not a natural candidate, and associations-based representations would avoid sampling. Particle filters are a mature concept and recent research focuses on initialization methods based on precomputation of sensor readings [6], [88].

In contrast to particle filters, Kalman filters [121], [22] and graph optimization methods [86], [135] rely on continuous probabilistic representations of the pose that are optimized for while taking uncertainty of detections and odometry into account. For Kalman filters, a Gaussian distribution is used to represent the robot pose and measurement error. Since the assumption of Gaussian distributions can be too restrictive, multi-hypotheses Kalman (MHKF) filters have been applied ([108], [68]) that can handle multiple distinct locations (resulting in a multi modal distribution). Graph based methods represent the poses of the robot (and possibly landmarks in the environment) by a connected network of nodes and edges, where the edges represent uncertain relations due to measurements, such as range measurements and odometry. A *factor graph* is a *hypergraph* that introduces the concept of a *factor* to relate more than two node variables to a measurement. Graph based optimization methods do not rely on the assumption of Gaussian uncertainty for the optimization variables. Instead, they often optimize for the *maximum a posteriori* (MAP) of the distribution (i.e., the most likely state given the measurements) without representing the distribution explicitly. Factor graphs are extensively used in SLAM [20] and incorporating uncertain data associations has been an active research topic [96], [122]. However, factor graph solvers are not guaranteed to converge to the global optimum and factor graphs are therefore still considered a *local* method that needs additional algorithms to handle highly ambiguous data associations.

All methods discussed so far rely on a notion of *detection* that can take many forms. For LiDAR based methods, both raw scan matching and feature based detection are popular. Iterative closest point (ICP) [133] is a popular method for determining relative poses between scans and between a scan and the map. On the other hand, feature based approaches first extract features from the sensor data (such as lines, corner, planes or cylinders) and relate those features to the map. The main benefit of features is that they can be *locally descriptive* and thereby less

sensitive to association error if the initial estimate of the feature location (based on a prior estimate of the robot location) is not accurate.

Finally, pattern recognition methods have become popular that work directly on sensor data by finding matches between local sensor data and a database of sample data that represents the map. Methods include bag-of-words based methods and machine learning approaches [48], [60], [7]. Place recognition methods have become popular in SLAM methods for loop closures and can also be used to initialize hypotheses for localization methods [10], [49].

In this thesis, the focus is on global localization with planar LiDAR which, in practice, is often solved by particle filters. A downside of particle filters is the limited control over the amount of particles needed and the lifetime of particles, which are obtained from sampling algorithms. Variants of MHKF have been suggested to deal with this problem more effectively, by generating hypotheses based on *creative features* found in individual laser scans and splitting hypotheses based on mismatch criteria [68], [4]. However, in cluttered environments this approach may also lead to generation of a large amount of hypotheses. Furthermore, many localization approaches rely on maps that have been made beforehand via SLAM methods, thereby representing the environment *exactly* for the SLAM platform. Generating SLAM maps can be time consuming, as a robot has to be manually operated throughout an entire building, but also can lead to the representation of temporary objects on the map. Global localization that is robust to disturbances and uses static object geometry representations of the environment is a topic that has received attention (e.g., [56]). However, both the representation and use of semantic maps for global localization and the addition of patterns to discriminate actively between clutter and structural objects remain relevant research topics; especially if they expand the domain of robots to larger environments.

1.2.4 Symbolics and semantics

Adding semantics to robotic world models has attracted much attention in recent years, both as a method for increasing robustness, and as a means of increasing reasoning abilities for more deliberated behavior (see, e.g., [46], [105], [77], [98], [25], [75]). A distinction can be made between symbolic representations that add *symbolic identifiers* to objects and relationships, and *semantic representations* that are concerned with the meaning of objects and relations. For example, adding a class label (e.g., *Door, Wall, Human*) to every point in a point cloud adds meaning (semantics) to the points, but is less symbolic than instance-level representations (in which we can, for instance count the number of unique doors). In [46], a distinction is made between *shallow* semantic representations and *deep domain knowledge* that links hierarchical (spatial) representations to general semantic knowledge. This semantic knowledge can contain information on topology and purpose of areas based on objects (e.g., a kitchen contains a fridge) from which inferences can be made. These inferences can then fill in the missing details when planning for a task. Reasoning capabilities have been researched especially in the domain of

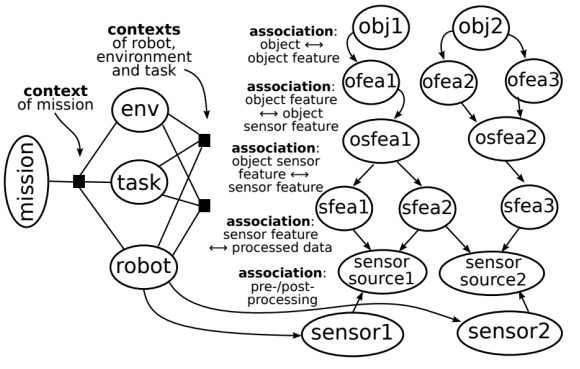


Figure 1.6. *An explicit view of the hierarchy as described in [18] that is used to relate sensor data to objects. This perception hierarchy is complementary to the control hierarchy that is used to perform actions related to objects.*

service robotics [46], [55], [125], [98]. In this thesis, we are most concerned with semantics that use this *symbolic anchor* as an argument in relations that connect to perception and motion algorithms (Figure 1.6). For instance, a column detection can be linked, via its circle representation, to a feature detection algorithm that tries to specify the type of column. Or it can be linked to a motion that takes sufficient safety distance into account. In this thesis, automated reasoning mechanisms are not considered; however, symbolic representations and their use for localization are. The main focus is on how we can exploit symbolic map representations by *domain-specific* decision processes such that they can be used by solvers (i.e., tree traversals and factor graphs) for localization. These decision processes are still largely *hardcoded* in the current demonstrators; however, they provide the symbolic handles that enable these kinds of reasoning mechanisms for increased deliberation and explainability. For our representations, we apply graph-based modeling concepts that are explained in [18].

1.2.5 Model-based and data-driven methods

Data-driven approaches have become very popular in robotics in recent years. The benefit of these methods over model-based approaches lies in the fact that they can learn parameters by optimization, using large data sets. Especially *neural networks* have shown great potential in detection and tracking tasks that seemed impossible before [24], [138]. These methods have also been applied successfully to robot perception, both as *front-end* detectors [34] and approaches that replace (parts of) the localization system [36], [134]. While effectiveness of these methods has been shown, explainability and generalization of the reasoning capabilities beyond training data is still limited [115], making it difficult to incorporate new knowledge regarding task, robot or environment. This thesis does not apply deep learning and instead focuses on model-based approaches. It is, however, important to note

the potential of combining learning approaches with the model based approaches suggested in this work. For example, deep-learning based image processing can provide semantically labeled detection as input for the local mapping strategy that is proposed, and data-driven optimization can potentially be of use in finding the model parameters, such as the probability that a detection is clutter. This discussion is left to the recommendations section.

1.3 Objectives and Contributions

The research contribution of this thesis to the state of the art is a methodology that enables and improves localization by adding relations to a map between objects and features, features and sensors, and feature patterns. This section further introduces the contributions for each of the three main chapters of this thesis.

1.3.1 Prior sources of world information

Making and maintaining maps for large indoor environments is a time consuming and error-prone task. Furthermore, scenarios exist where making maps beforehand is not possible. This makes the use of external sources of information attractive, especially if they are standardized and widely used. Chapter 2 considers the use of Building information Modeling (BIM) representations for localization. BIM models are widely used in architecture and construction to exchange semantic and geometric data of a building. Furthermore, they play an increasingly important role in the maintenance of digital twins of buildings [116] as up-to-date representations for facility management. The objectives of this chapter are the following:

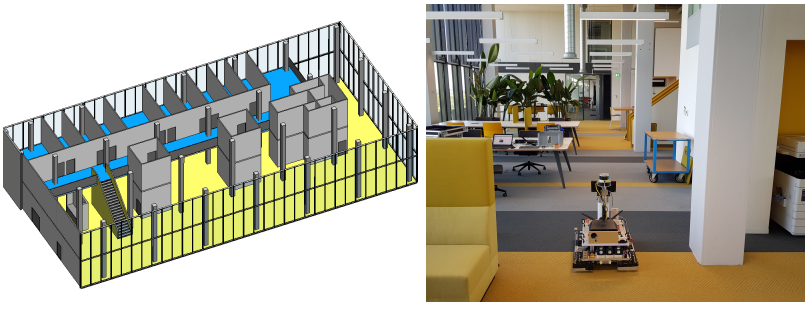
1.1 *Convert the entities of a BIM model to a representation that contains explicit object-features for use in robot localization*

1.2 *Implement a pose tracking algorithm for a robot equipped with planar LiDAR that maintains data associations with these object-features.*

The first contribution of this chapter is that we propose a strategy to convert walls and columns from a BIM model to a graph-based world representation. This representation links BIM entities to geometric features for a specific sensor. The second contribution is a moving window factor-graph localization algorithm that extracts line segments and corners from planar LiDAR data and associates them with these static building features. The approach is demonstrated using a BIM model of part of the ATLAS building (Eindhoven University of Technology) in which the robot pose is tracked by querying local geometry from a spatial database (Figure 1.7).

1.3.2 Association-based localization

In Chapter 3, we consider global localization, which is a relevant capability for a mobile robot in order to recover and continue its task autonomously. We propose



(a) (b)

Figure 1.7. *Tracking experiment in the Atlas building using a BIM model to supply the perceivable geometry.*

an answer to the second research question, by determining when and how a robot should associate its sensor data with the map. The objectives of Chapter 3 are:

2.1 *Maintain hypotheses on associations between local features that are extracted from sensor data and features on the global map*

2.2 *Score and prune the hypotheses in order to retrieve the correct robot location, while being robust to objects that are not on the global map.*

2.3 *Compare the results with a Monte-Carlo method that relies on sampled pose-based hypotheses and a grid-based representation of the world.*

The contribution of this chapter is a solution that splits the localization problem into two activities: 1) Maintaining a local map in the vicinity of the robot with features that can be reliably associated *locally* and 2) Evaluating the global associations whenever a new local stable feature enters the local map. The approach associates local *measurements* into *local features* and associates local features to global features on the map using association hypotheses, resulting in an association hierarchy (Figure 1.8). Location hypotheses are maintained by explicitly representing a set of plausible local-to-global associations, which we evaluate by expanding a hypothesis tree. The lack-of-fit is assessed using a horizon of these features that includes the new candidate association. The method does, therefore, not rely on Gaussian or randomly sampled belief approximations for the pose. The benefits of this approach as compared to particle filter approaches is that it quickly initiates a much smaller set of plausible hypotheses based on the local map, which decreases the amount of hypotheses needed and increases robustness. Furthermore, the approach is *symbolic* in its associations and uses unmatched local features to prune unlikely hypotheses. To show the performance of the approach, we run experiments in the model that we introduced in the previous chapter, which is manually corrected for geometric deviations. We add disturbances to the environment in some of the

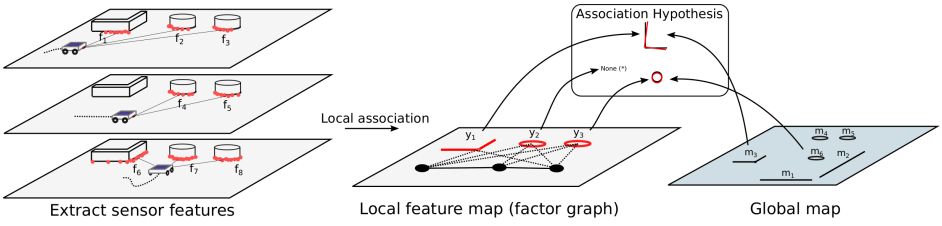


Figure 1.8. *The approach suggested in Chapter three.*

test scenarios and compare our method with the popular ROS AMCL particle filter implementation, to show the advantages and trade offs of each method.

1.3.3 Patterns and association hierarchy

In Chapter 4, we further extend the association hierarchy to incorporate patterns into the local map. A major challenge in localization is the differentiation between structural building elements and false detections or items that are not represented on the map. Taking into account unmapped items results in a rapid growth of the number of hypotheses that must be maintained between the local and global map. Therefore, the objectives of this chapter are:

3.1 *Add a layer of geometric pattern relations to a map* a priori *that can help a robot to differentiate between clutter and structural building features.*

3.2 *Extract these patterns from local sensor data and exploit them while making data associations.*

3.3 *Disambiguate patterns in the case of repeating structure, by incorporating local motion primitives and the absence of expected features.*

The first contribution of this chapter is that we add a layer of collinear and right-angled pattern relations found in concrete columns to the map. The rationale behind this approach is that indoor environments often exhibit a distinctive grid-like structure (Figure 1.9) The second contribution is that these geometric constraints are used in the local map to find pattern candidates which are then prioritized in the making of data associations, by using their relative pattern dimensions. The third contribution is that the ambiguity found in these patterns is dealt with by performing a local rotation action while evaluating an association tree that is initialized with the pattern. In the evaluation of this tree, we use all matching primitive features in the environment of the pattern, as well as a free space check to effectively find which pattern on the global map is supported. The method is experimentally demonstrated using a closed loop autonomous robot, showing both initialization and recovery behavior using patterns.

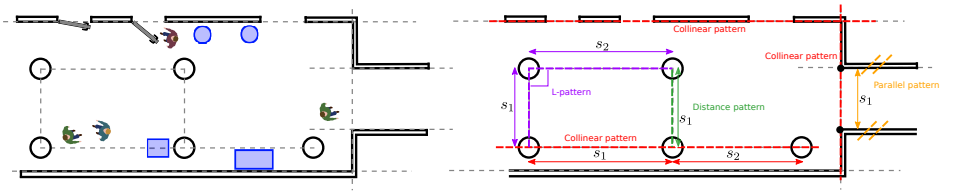


Figure 1.9. *Discriminating patterns that occur in indoor environments and can be exploited.*

1.4 Research Project

The work in this thesis has been carried out as part of the New Frontiers in Autonomous Systems (FAST) project. The FAST project consists of a consortium of five private companies (Vanderlande, Lely, EXRobotics, Rademaker and Diversey) and Eindhoven University of Technology. The goal of the project is to develop semantic world models for mobile robots that enable increased flexibility in industrial scenarios as well as socially aware navigation capabilities. The project was carried out collaboratively by a multidisciplinary team of four PhD candidate researchers and company R&D engineers. The project received additional funding from TKI-HTSM [62].

1.5 Outline

The main chapters of this thesis are based on research results that have been published or submitted for publication. Starting with Chapter 2, the application of BIM models for localization is introduced and demonstrated by experiments in a campus building. Next, Chapter 3 considers global localization in the same environment, based on associations between a local and global map. Chapter 4 elaborates on this approach by introducing and evaluating feature patterns in a second indoor scenario. Finally, Chapter 5 summarizes the relevant conclusions considering all main chapters and provides a final discussion with recommendations for future work.

Chapter 2

Connecting Semantic Building Information Models and Robotics

Abstract
This chapter proposes a method to integrate the rich semantic data-set provided by Building Information Modeling (BIM) with robotics world models, taking as use case indoor semantic localization in a large university building. We convert a subset of semantic entities with associated geometry present in BIM models and represented in the Industry Foundation Classes (IFC) data format to a robot-specific world model representation. This representation is then stored in a spatial database from which the robot can query semantic objects in its immediate surroundings. The contribution of this work is that, from this query, the robot's feature detectors are configured and used to make explicit data associations with semantic structural objects from the BIM model that are located near the robot's current position. A graph-based approach is then used to localize the robot, incorporating the explicit map-feature associations for localization. We show that this explainable model-based approach allows a robot equipped with a 2D LiDAR and odometry to track its pose in a large indoor environment for which a BIM model is available.

This chapter is based on:
R. W. M. Hendrikx, P. Pauwels, E. Torta, *et al.*, "Connecting Semantic Building Information Models and Robotics: An application to 2D LiDAR-based localization", in *Proceedings - IEEE International Conference on Robotics and Automation*, 2021, pp. 11 654–11 660

2.1 Introduction

Mobile robots are deployed more and more in dynamic environments shared with and familiar to humans such as hospitals [109], restaurants [118] or nursing homes [27]. Due to the familiarity of the environment, there is an expectation that mobile robots will not only robustly perform long-term autonomous tasks, but that they do so using the same semantics that operators, bystanders and engineers use. To this end, adding semantics to existing geometric representations used for localization is an extensively researched topic. We present a case study for leveraging an existing standard and available models for semantic building representation from the built environment domain for indoor localization. These models, when available, can provide an alternative to robot-specific map creation. In this study, we use a robot equipped with a conventional 2D laser range finder, which is still to be found on many existing platforms. We show how semantic building information models can be used for explainable 2D LiDAR based localization in environments where other sensor modalities may be infeasible due to cost or privacy-by-design requirements.

2.1.1 Contribution and Contents

In our work, we identify the following contributions:

- Proposing a workflow to leverage the semantic and geometric information in building models for indoor robot localization, via composition with a robot-specific property-graph representation in a spatial database.
- Demonstrate the feasibility of the approach for a location tracking task based exclusively on static semantic building feature associations.

The chapter starts by discussing related work from both the robotics domain and the built environment domain in the next section. Hereafter, we introduce the property graph approach for world representation, followed by the step of populating this world model with BIM entities. This is followed by a concise treatment of our localization approach, which uses existing graph optimization techniques to obtain the robot pose from semantic data associations. In section V, an indoor experiment is then presented in which a robot localizes indoor based on a BIM model. Finally, the technical challenges that arise when using existing building models for localization are discussed, and topics for future work are identified.

2.2 Related Work

Semantic indoor maps for robotics

Among the various representations for indoor semantic maps used in practice (see, e.g., [77] and [83] for a review), we focus on object-oriented representation standards that have been used for robot localization. OpenStreetMap (OSM) is an

opensource crowd-sourced mapping initiative that has been extended for indoor robot navigation in [92]. The authors propose an indoor tagging schema to represent the indoor domain hierarchically using the OSM primitives. Objects such as walls and doors are tagged, and related to rooms and hallways. Furthermore, specific areas are represented and tagged to define, e.g., traffic lanes for navigation. The authors show an example of a hospital environment that was modeled and tagged manually using architectural drawings as a reference. The authors acknowledge that the manual creation of the maps using the available Geographic Information Systems (GIS) tools is tedious and that the choice of georeferenced (latitude/longitude) coordinates is not obvious for indoor geometry. Another important initiative for building modeling is indoorGML [74], which specifically targets indoor spaces and represents geometry, topology and semantics in a multi-layered space model. While indoorGML provides many relevant modeling concepts for indoor navigation, existing work is limited to automatic extraction of indoorGML models from occupancy grid maps in [126]. A different representation of semantic maps in spatial databases is suggested in [31] in the form of the SEMAP framework. In their work, the authors represent semantically annotated 3D object models in a PostGIS database. The models are geometrically represented in PostGIS' own 3D extension of the "Simple Feature Access" [120] specification with the addition of a spatial ontology for map queries. The authors show multiple applications, including topological location queries of the map for an agricultural harvesting scenario. The authors do not focus on metric robot localization from on-board sensor data. Another work that has similarities to ours is [14], in which the authors perform robust indoor localization with a laser scanner starting from architectural floor plans. They augment the floor plans using pose graph SLAM techniques and robust matching criteria. This makes their approach able to perform long-term navigation in the presence of map mismatches and environmental disturbances. Contrary to the approach we present, the authors do not focus on semantics or standardized map formats in their work. Furthermore, they propose a method to provide an updated scan-based map that is consistent with the prior floor plan and contains all geometry, which is not within the scope of our work.

In the above works, there is a strong reliance on maps with a geospatial background (e.g. GML, GIS), or architectural floor plans. These sources are not widely available, nor fully reliable. Yet, for many buildings, detailed Building Information Models (BIM) ([35], [16]) are currently available, which include 3D geometry as well as detailed semantic data (materials, object properties, etc.). In this work, we want to take advantage of the increasing availability of building information models as *digital twins* to propose an alternative approach to semantic map creation and representation for the robotics domain. We do so by creating an explicit link between the semantic information contained in Building Information Models (BIM) and robotics semantic maps. Our approach allows to automatically populate a semantic map, avoiding the manual effort cited by [92]. In addition, the robot's semantic map and the BIM model use the same semantics and the same reference coordinates to indicate features in the environment, which we consider an im-

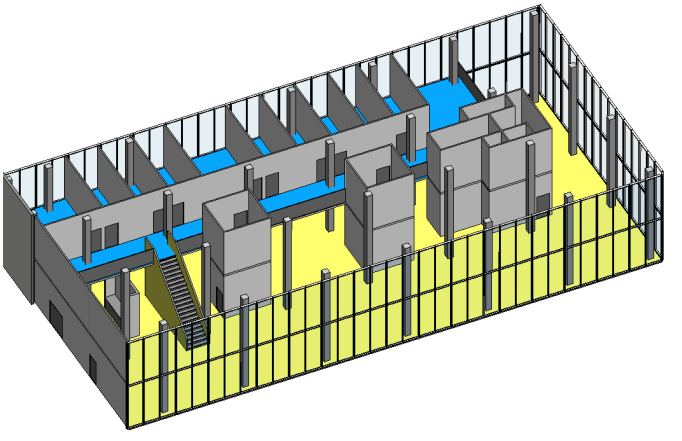


Figure 2.1. *Example of the geometry present in a BIM model (here shown as part of a storey). The model shown is part of the ATLAS building at TU Eindhoven and is used in this work. It is extracted from a larger BIM model.*

portant step in reaching shared semantic understanding of indoor environments between humans and robots.

BIM models

3D BIM modelling software and BIM modelling processes are increasingly taking over the Architecture, Engineering, and Construction (AEC) industry. In many countries, newly built buildings are modelled in BIM software, and delivered to the client as an as-built model. The operational phase of the building predominantly relies on Facility Management Information Systems (FMIS), which typically have considerably less detailed 3D geometric data, and instead focus heavily on the collection of sensor data (access, temperature, air quality, ventilation, etc.). This collection of data is often termed 'digital twin': the digital counterpart of the physical building. A dominant data standard in the AEC industry is the Industry Foundation Classes (IFC) [65]. This standard focuses heavily on the interoperable exchange of 3D data across BIM authoring tools. An open and neutral IFC file can be exported from a BIM modelling tool, making semantic and 3D geometric data openly available (human- and machine-readible). Although this data source is used less often for existing buildings [132], it still provides an invaluable resource of information if it is available. Whereas IFC has always been available in the EXPRESS information modelling language, recent works have aimed elaborately at enabling XML, JSON and RDF formats for the same data. An XML format has been supported since the early 2010s, the RDF format for IFC data is available since 2016 [100], and a simplified JSON format is under construction at the time of research and writing [64].

Previous research has tried to leverage BIM models to tackle the challenge of path planning and localization. With respect to localization, BIM models have been mostly used for indoor image-based localization. Acharya et al. [1] generate a data-set of synthetic images with associated known 6-DOF camera locations and orientations from BIM models. The synthetic data-set is used to train a Deep Convolutional Neural Network (DCNN) which is used for indoor localization. Similarly, [54] generates a data set of synthetic images from the BIM model and trains a DCNN to extract features from the generated synthetic data set. Features extracted from the synthetic data are compared with features extracted from images of the physical environment and used to select the synthetic image (with known 6-DOF pose) that is closer to the physical image. Both works differ from what we present in this chapter because the localization method is based on camera and a DCNN that has to be trained on the specific building model. Additionally, these papers do not focus on describing a methodology to automatically extract such information from the BIM models. Our approach differs because (a) the BIM model directly provides the features to look at without the need for training the DCNN and (b) we present a methodology to automatically extract relevant information from the model avoiding much of the error-prone manual effort. Other work has focused on using the BIM model to derive the topology of an indoor environment from which a path can be planned. In this case, localization is not further elaborated, as opposed to what we aim at in this article. In [114], the authors propose to extract information from BIM models to set-up a simulation environment (VEROSIM) for robotics development. The environment can connect the OMPL (Open Motion Planning Library) to the imported BIM model to generate collision free paths. On a similar line [97] derives a topological graph from BIM models upon which an A* planner can retrieve the optimal path. Although they are valuable and important reference works, our work aims precisely at a live localisation based on a current model of the building model and matching of geometric features.

2.3 Connecting Bim Data to the Robot's World Representation

2.3.1 World model representation

We use the term *world model* to describe the robot's internal representation of itself and its surroundings (i.e., the map) in relation to its sensors. To accommodate this representation, we use a property graph data model (Figure 2.3) which enables composition of different domain models. This general data model has been used extensively in knowledge representation for robotics (see [98] for a review). In this work, we focus only on the semantic map and its relation to available sensors. To represent the property graph we use the JSON-LD host language, which provides the mechanisms for attaching a unique symbolic id (`@id`), model id (`@type`) and namespace reference (`@context`) to every entity in the world model. For the geometric

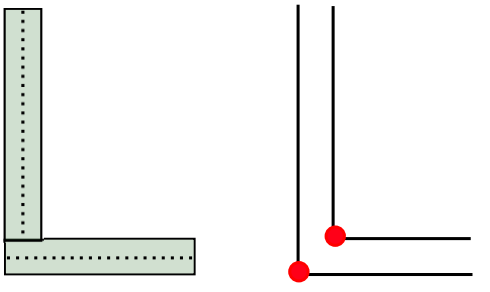


Figure 2.2. *Conversion of the BIM line-with-offset representation of a wall (left) to a representation consisting of two polylines for each wallsegment with a shared Point on the corner connecting two wallsegments (right).*

primitives in this work, we use 2D "Simple Feature Access" [120] representations. We augment these primitives by giving each point that belongs to a composition (such as polygons) an individual `id`, allowing to maintain topological consistency (e.g. walls that share a point which resembles a corner). These geometric representations are then associated with the semantic entities by `represented_by` relations, allowing to loosely couple different representations when desired. Furthermore, we explicitly relate representations of objects to the sensors through which they can be perceived. We introduce the `ObjectFeatureRepresentation` relation that connects the geometry representation, the object and the sensor, allowing the robot to query for features that it can perceive using its sensors.

The property graph model for localization has to be generated from the BIM model for all relevant objects. This procedure is schematically depicted in Figure 2.5. First, the BIM model is queried for relevant objects, which can appear at different positions in the model hierarchy (e.g., an `IfcWall` can be declared as part of a space, or as a connected attribute of another `IfcWall`). For this reason, the BIM model of a floor of the building is exported to an IFC-JSON representation [64] which is then made into valid JSON-LD by adding a `@context` specifier for the types and relations. This JSON-LD representation can then be "framed" by the JSON-LD API [73], turning it into a tree structure where objects of interest are at the root for processing without requiring a graph database. In this chapter, we use the `IfcColumn` and `IfcWall` entities from the IFC model for localization. These entities are queried from the model together with their `representation` and `objectPlacement` relations. For the columns, the `sweptArea` 2D representation is converted into a polygon profile, for which the local coordinates are converted into global coordinates using the column's `objectPlacement`. For the walls, an `IfcPolyline` represents the center line, together with an offset for the thickness. This representation is converted into a slightly different representation with two (inner and outer) polylines, that connect to adjacent wall segments using corner points (see Figure 2.2). This preserves the topology of wall segments and corners. The final result is exported as a JSON-LD property graph, which is partly visualized

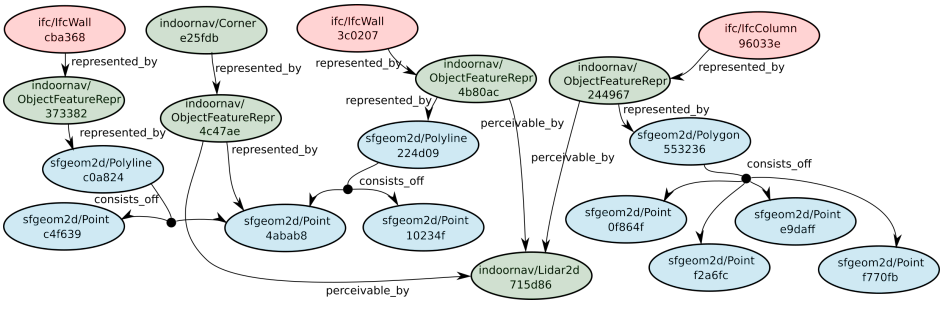


Figure 2.3. *Graph representation of the semantic entities from the BIM model and the geometry representations that are perceivable by the sensor, in this case the 2D LiDAR. Different domains (i.e., simple feature geometry, IFC entities and navigational relations) are shown in different colors. The id of entities that are not part of the BIM model are randomly generated.*

in Figure 2.3. A small section of the resulting map of a floor of the building considered later in this work is represented in Figure 2.4.

2

2.3.2 Spatial database and queries

For querying the spatial features and their semantic relations, we use the well-known PostgreSQL database with the PostGIS spatial extension. We store the property graph representation in the database as well, making it possible to query for relations and entities using the SQL query language. The database is queried for spatial features that are close to the robot. The sensor type is part of the query, resulting in features that are part of an `ObjectFeatureRepresentation` perceivable by the given sensor. The query returns the feature id, feature type, object id and object type for each feature, together with the spatial object that contains the actual coordinate data structure. For example, a query for perceivable features with a 2D LiDAR near the current position, may return the object `{type:"IfcColumn", id:"96033e"}`, together with its representation `{type:"Polygon", id:"553236"}`. This explicit symbolic link between the geometry, its interpretation and the object will be maintained in the association-based localization approach.

2.4 Localization

While the robot is tracking its location, it queries semantic map features from the database that are perceivable by its on board sensors and are currently within a certain radius around the robot's location on the map. We refer to these as *map features*. Our approach then tries to match them with features found in its sensor data, so called *sensor features*. This map-query-first approach allows to extract only

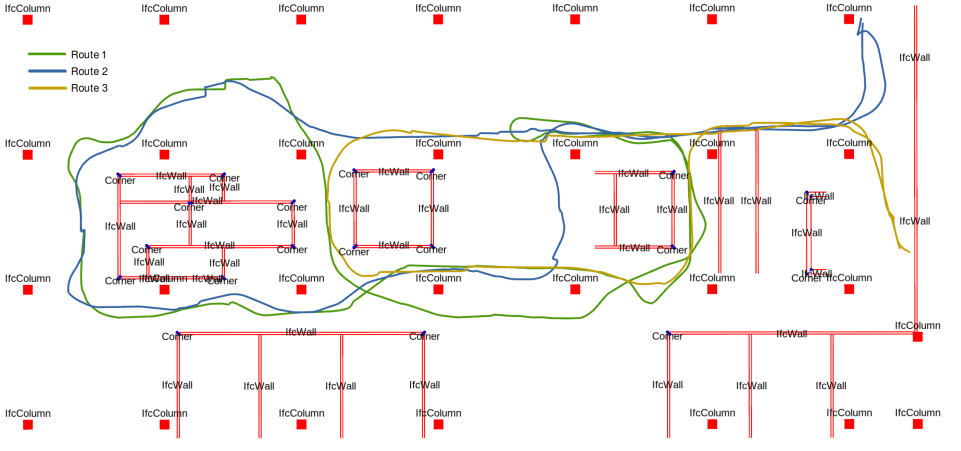


Figure 2.4. *The map considered in this work, as generated from the BIM model, with static features relevant for the LiDAR sensor. The features are annotated with the types of the objects they represent. The paths driven by the robot are shown, as determined by our localization approach, starting at different positions. All three paths finish at their respective starting positions. The final trajectory was recorded while three actors were actively obscuring the laser field of view. Notice that at some points small jumps occur in the map position, because of new associations with structural elements of the building.*

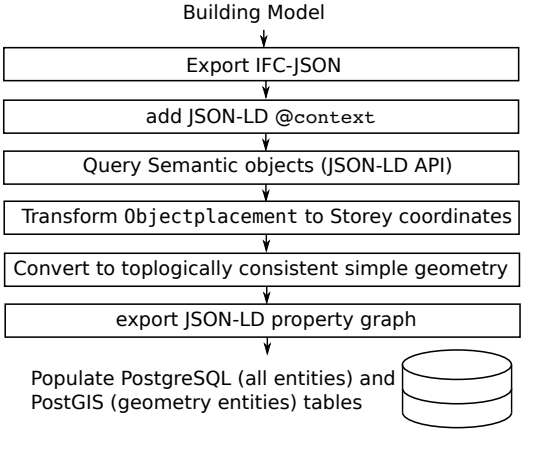
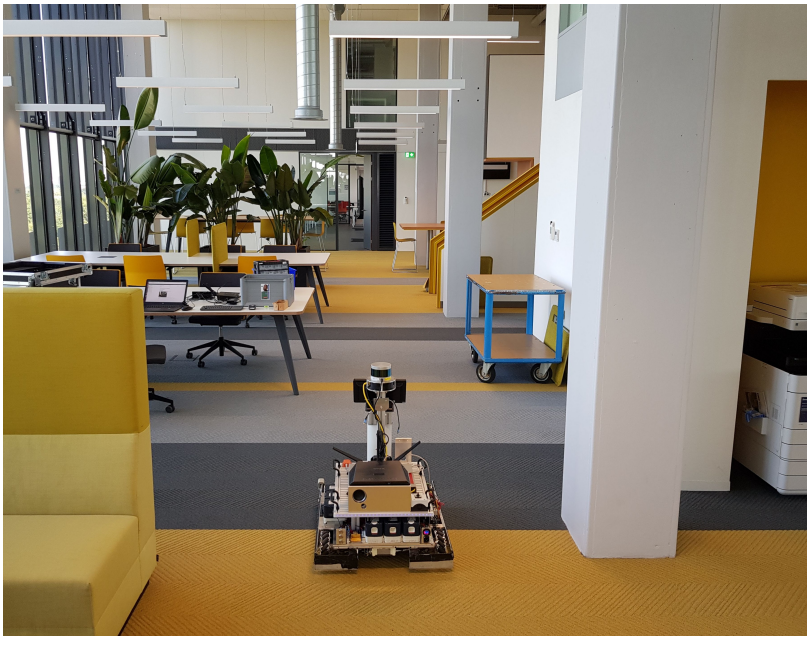
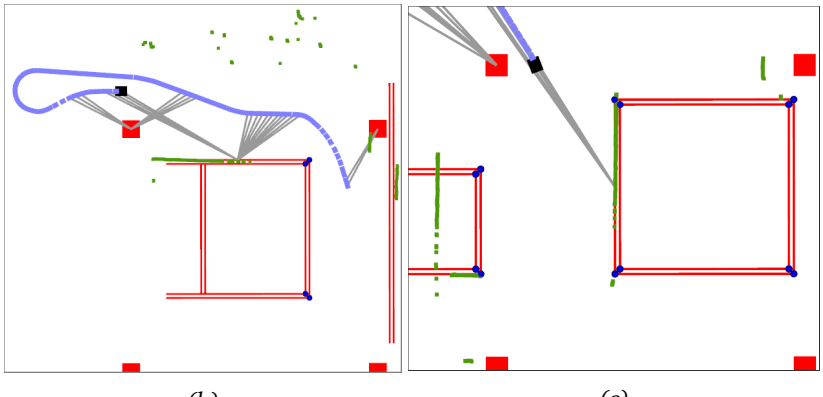


Figure 2.5. *Conversion from the BIM model to the representation used by the robot, stored in a database, as used in this work. This database contains both the property graph entities and the geometric entities (using PostGIS).*



(a)



(b) (c)

Figure 2.6. *(a) Picture of the indoor environment with the robot. (b) The same location showing the localization output. Matched semantic features are indicated by grey lines originating from their corresponding robot pose. LiDAR points are in green. (c) An example of the mismatch between the BIM model and the actual environment, making localization more difficult. The location of the square space showed a significant mismatch with reality, causing the robot location to jump while making correct associations. Furthermore, due to glass and doors, not all walls are always perceivable.*

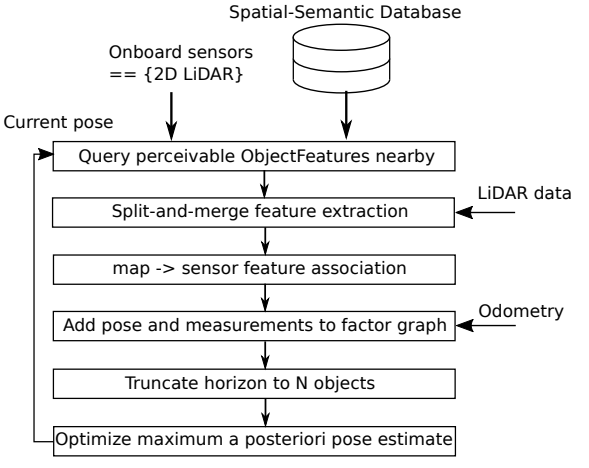


Figure 2.7. *Visualization of the localization approach that queries features from the database, matches sensor features to them and optimizes the resulting factor graph.*

sensor features that are currently of interest. The current implementation supports the extraction of line features, corner features and box features from the sensor data, based on a split-and-merge weighted line fitting implementation from [102], [47]. The corner and box features are obtained by checking if the lines support the well-known L-shape used often for rectangular objects. In the configuration of the detectors, we demand that the line segments used for wall and corner detection are sufficiently long to be insensitive to open doors. While this threshold is currently manually set to 1.2m, it could be derived from the `IfcDoor` representations in the BIM model if these are present.

The measurements are added to a factor graph containing the robot poses over a variable horizon, as well as range-bearing measurements to perceived objects. For columns and corners we use the well-known range-bearing measurement model. For walls, we use a model that constrains the angle and distance to the wall, but not the position alongside it. If a match in the sensor data is found based on the features suggested by the map (e.g., a line segment that falls within the line segment of the wall representation up to a threshold), the feature and measurement get added to the factor graph. We explicitly reference the `id` of the features in the factor graph, making the data associations with the map explicit. This feature-based approach has benefits over scan matching based approaches such as ICP, because it first checks if the sensor data supports the primitive feature suggested by the map to be usable. This also allows to disable individual features without removing them from the map by modifying their *perceivable_by* relation.

Since the semantic features we use can be relatively sparse, substantial drift corrections are possible. We use graph optimization to avoid inconsistencies due to linearization. We do not focus on the trade offs of graph-based mechanisms for pure localization in this work (See, e.g. [136], [135] for a comparison of factor

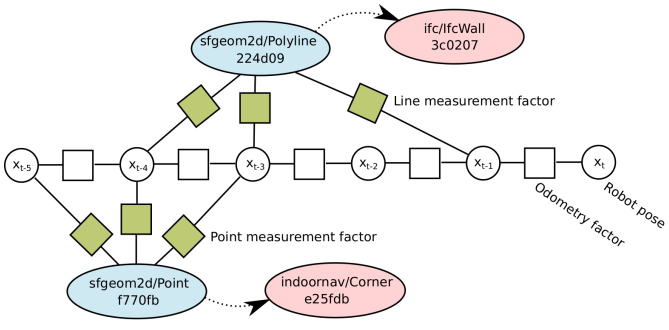


Figure 2.8. *Example of a factor graph containing the features and measurements within a certain horizon of robot poses.*

graphs and particle filters for pure localization). Our main focus is to show that this method makes observations explicit. Exploiting these explicit links for robust navigation tasks or updating the BIM model real-time using SLAM is potential future work. We use a moving-horizon approach in which the horizon is adjusted based on the number of geometric features that have been uniquely associated with the map. When more than N of these unique geometric map features are present in the horizon, we remove the oldest features and their corresponding measurements until again N features are present. We have implemented the localization in C++ using the GTSAM [32] optimization library together with our own in-memory semantic graph model and bookkeeping functions.

2.5 Experiment with a Robot Platform

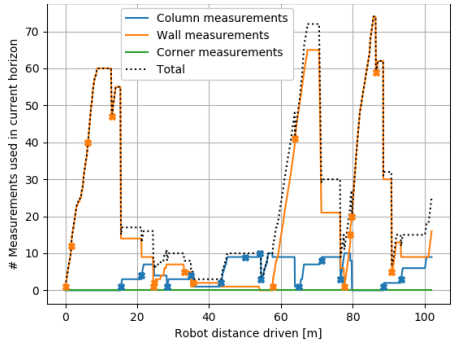
In this section, we show an experiment with a mobile robot on a floor of the building that is shown in Figure 2.1 (Atlas building, TU Eindhoven) for which an IFC model is available. We use a custom-made platform (Fig. 2.6a) equipped with mecanum wheels, wheel encoder odometry and a Hokuyo UTM30-LX 2D LiDAR scanner mounted upside down, close to the floor. Due to its mounting position, we have a 180° field of view consisting of 720 scan points. We teleoperate a total of three routes (approx. 100 meters each) with the robot, starting from different initial poses. The latter are provided manually to our algorithm by initial pose estimates. The environment is cluttered with both semi-static objects (furniture, plants) and dynamic objects (chairs, carts) which are not present in the semantic map used for localization, as shown in Figure 2.6a. In the third route, three actors are walking around in the field of view of the robot. The robot's feature detection is triggered whenever a distance of 15 cm has been driven and the map is then first queried for features that are visible to the 2D LiDAR within a range of 6 meters. The sensor data is then processed to search for sensor features that support the object

features on the map and the pose estimate is updated. Figure 2.4 shows the 2D map with the features perceivable by the 2D LiDAR and the three trajectory estimates resulting from our localization approach. The horizon length is truncated at all times based on three semantic object references. Figure 2.6b shows an example of the horizon in which these references are visualized by edges (grey lines). Using our approach, the robot was able to successfully track its pose by incrementally associating perceived features with objects on the map generated from the BIM model. In the next section, the results are discussed in more detail.

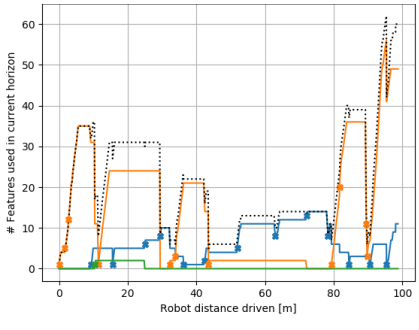
2.6 Results and Discussion

The amount of measurements associated with the building model in the horizon at that timestep are visualized in Figure 2.9, grouped by object type. Both the number of associated measurements with an object of that type and newly detected objects entering the horizon are depicted (the latter by markers). The resulting trajectories are shown in Figure 2.4. The object types (and their features) used for localization in our approach were selected because of their availability in the BIM model and their saliency. No false positive association was made by the localization algorithm in our experiments. We selected these features because we predict that their saliency can carry over to indoor environments different than the one we targeted. In Figure 2.6 it can be seen that the wall features are very salient: once detected, they have a high recall and are spotted multiple times within the horizon. The column features are consistently spotted as well, but because of the L-shaped visibility requirement, they are only detectable from certain positions with respect to them. Again, this approach was deliberately chosen to favor precision (i.e., minimize the occurrence of false positives). The corner features are spotted less frequent because of the same L-shape detection requirement. Spurious corner measurements can give rise to inconsistent matches and are avoided by this stringent shape requirement. Furthermore, from Figure 2.6 we can see that columns and walls are both necessary for consistently having recent features within the horizon. These features are to be found in many buildings, making our approach applicable in many cases where a sufficiently rich BIM model is available. However, we emphasize that also the semantic explainability of local detections is an important feature of our work, that will come to its full right when exploited in context of different robot tasks, such as navigational rules (e.g. driving close to walls or using a column as natural waypoint) and when interaction between humans (e.g., facility managers) and robots will take place.

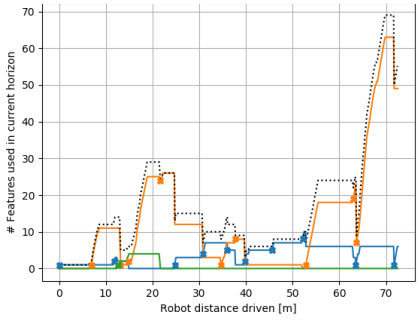
Another observation from our experiments is that the BIM model is not always accurate or complete. Spatial inaccuracies were present, one of which is shown in Figure 2.6c. Although we did not deal with these inconsistencies explicitly (recovery mechanisms are discussed in the next chapter), our method is able to make the right associations in the considered building model. We do note that robustness against unmodeled dynamic clutter is not incidental, because we focus



(a) Path 1



(b) Path 2



(c) Path 3

Figure 2.9. *The amount of feature measurements in the horizon used at each time step for pose optimization, grouped by the object type of the feature (note that there is overlap until an object gets removed from the horizon). The markers indicate a new object entering the horizon, after which the objects is often spotted multiple times, increasing the measurement count. The horizontal axis is discrete, taking steps of 15 cm (the update trigger distance).*

on features that are known to be static for localization. A final remark is that the relative inaccuracy of the BIM model raises an important question about the definition of accuracy. Whether we want to define accuracy as the metric deviation of the position in a map coordinate system, or as the correctness of perceiving local objects of interest with respect to the robot remains a relevant question. The latter enables a definition of accuracy in the case where supplied maps are not perfect but semantic navigation based on correct associations can be robust nonetheless.

2.7 Conclusions and Future Work

In this chapter, we showed that existing semantic building information models can provide a robot with enough information to localize itself, providing a great opportunity for automatic deployment in large buildings without prior work or adaptation. We also showed how this semantic information can be translated to explainable associations, used for localization by a robot equipped with a 2D LiDAR, using a property graph database to let the robot query its semantic environment. The opportunities we see for future work will consist of generalizing our approach to different semantic entities that are present in BIM models (such as doors or curtain walls) and using different sensors to perceive them. Furthermore, keeping the BIM map up-to-date and consistent is an important challenge, with great potential to be of use for maintaining digital twins in the operational phase of buildings. To conclude, we foresee that automatic deployment of robots in buildings can be useful in many scenarios and our work has explored important steps in making this a possibility.

Chapter 3

Local-to-Global Hypotheses for Robust Robot Localization

Abstract
Many robust state-of-the-art localization methods rely on pose-space sample sets that are evaluated against individual sensor measurements. While these methods can work effectively, they often provide limited mechanisms to control the amount of hypotheses based on their similarity. Furthermore, they do not explicitly use associations to create or remove these hypotheses. We propose a global localization strategy that allows a mobile robot to localize using explicit symbolic associations with annotated geometric features. The feature measurements are first combined locally to form a consistent local feature map that is accurate in the vicinity of the robot. Based on this local map, an association tree is maintained that pairs local map features with global map features. The leaves of the tree represent distinct hypotheses on the data associations that allow for globally unmapped features appearing in the local map. We propose a registration step to check if an association hypothesis is supported. Our implementation considers a robot equipped with a 2D LiDAR and we compare the proposed method to a particle filter. We show that maintaining a smaller set of data association hypotheses results in better performance and explainability of the robot's assumptions, as well as allowing more control over hypothesis bookkeeping. We provide experimental evaluations with a physical robot in a real environment using an annotated geometric building model that contains only the static part of the indoor scene. The result shows that our method outperforms a particle filter implementation in most cases by using fewer hypotheses with more descriptive power.

This chapter is based on:
R. W. M. Hendrikx, H. Bruyninckx, J. Elfring, *et al.*, "Local-To-Global Hypotheses for Robust Robot Localization", *Frontiers in Robotics and AI*, 2022

3.1 Introduction

Localization is an essential part of an autonomous mobile robot system. The absence of global position sensors and the presence of map disturbances results in challenges that are specific for indoor scenarios. There is a clear trend of deploying robots in indoor environments where they have to be able to robustly deal with changing environments, such as restaurants [118], hospitals [109] or nursing homes [27]. This raises the expectation of robotic systems on multiple fronts. First, robots are expected to leverage geometric and semantic information from existing sources, which are already available for many indoor environments, such as semantic building information models. This prior knowledge is sensor-independent and can constitute building geometry (walls, corners, columns, doors) or topological information such as room numbers. Second, robots are expected to not only track their pose in a building, but to re-obtain their location quickly after failure or reset, without an operator intervening. In the previous chapter, we explored obtaining maps from industry standard building models which are already used to share building data [59]. In this chapter we focus on global localization, proposing a method that deals with the ambiguity and uncertainty in the environment on an association level. We show that exploiting local structure first makes the localization outperform a gridmap-based particle filter, while also making it inherently more symbolic and thereby semantically insightful and configurable.

3.1.1 Requirements and scope

We focus on indoor localization in common public environments. The following requirements have led to this work:

- An existing global map must be available with high-level features for the sensor that is being used, which are explicitly linked to semantic instances of static indoor features.

- The robot system consists of an odometry sensor and a sensor from which features can be extracted that can be associated with the global map features.

- The robot must be able to recover its global location in the environment by making explicit association assumptions, while being robust against stationary *unmapped objects*.

The first requirement assumes that feature instances (e.g., walls, columns) are linked to sensor representations (e.g. lines, corners) as described in [59]. In this chapter we use a system equipped with wheel encoder odometry and a conventional 2D planar laser scanning device. We assume that static environment features are resembled well by mostly straight or circular geometries on an existing feature map.

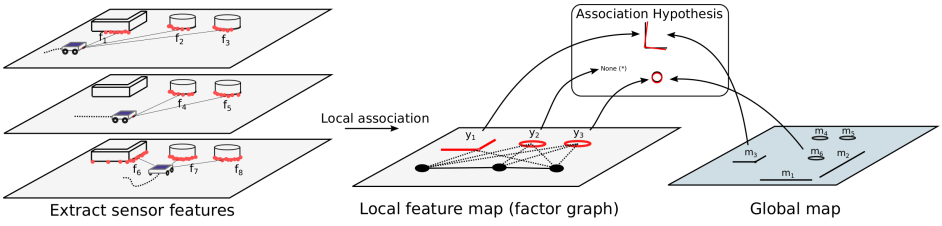


Figure 3.1. *A graphical depiction of the method we propose in this chapter. A single hypothesis is shown as an example.*

3.1.2 Proposed method

Our method localizes globally by first combining local *sensor features* (geometric features such as lines, corners and circles) from separate scans into a consistent *local feature map*. These *local features* are then associated with *global map features* using an association tree. The levels of this tree represent the local features, and the nodes represent the possible association of the local map feature with a global map feature. We evaluate association likelihood based on a moving horizon strategy and prune unlikely associations based on feature description and spatial congruency, leading to a hypothesis tracking approach where the sample space is that of data associations. Note that our method is sensor-independent and well-suited to function with different or multiple sensors modalities. Our methods deals with *unmapped objects* on a feature association basis and does not rely on a map containing all visible geometry. We assume that the features on the global map are geometrically accurate and that objects that have velocity do not resemble our static features. The method is graphically depicted in Figure 3.1.

3.2 Related Work

Dealing with multiple hypotheses, one can distinguish between location-driven and feature-driven localization approaches as remarked by [4]. Location driven approaches maintain hypotheses using a sample set on a continuous space, or use an a-priori selected topological graph in which the nodes represent locations in the environment. On the other hand, feature-driven approaches explicitly maintain hypotheses on associations between features that are seen locally and features on a map. The latter characterizes our method; however, relevant prior work exists using both approaches which we will now consider.

3.2.1 Hypothesis tracking methods

A reference work for our method is [4], in which the authors perform global localization in an environment described by geometric (line and point) models. They maintain a tree of local-to-global data associations together with multiple

extended information filters (EIF) to solve for the map location of the robot. They employ both binary constraints to check if feature pairs are metrically consistent both locally and globally and, when possible, use rigidity constraints to check the likelihood of a new pairing given the location based on old pairings. Rather than a Bayesian approach, they use an approach that designates hypotheses as equally valuable options, and use a *lack-of-fit* measure only in their duplicate removal strategy. They also use an explicit clutter hypothesis and verify their approach in an experiment. [56] expand on this work by introducing a relation table that captures the coordinate independent local structure of a map of line segments in a more efficient way. They check the correspondence of line segments (congruency under rotation and translation) on a local and a global map by introducing a geometric relation comprising two lengths, an angle and two relative distances between the segments. Using this relation they propose depth-first constraint-based search in an interpretation tree to find matching line segments for localization without a prior pose being available, and account for an *unmapped item* option (*null features*). Their work first applies pair-wise constraints to balance the complexity of the search in the tree and afterwards checks an alignment constraint of the total map based on a least-squares solution. They limit the total number of *null features* allowed in order to mitigate search time, and mention that determining this parameter is environment-dependent. The authors also use a topological representation (e.g., hallway, room) of their environment with annotated visible line segments for each location, in order to restrict relation matches to simultaneously visible pairs. They provide a comparison of their method and Monte Carlo Localization (MCL) in a lab environment with clutter objects added and show increased performance. Our method draws inspiration from certain aspects of the two mentioned works, as we will elaborate on later. Other works for different sensor modalities such as [87] provide a camera-feature based multi hypothesis approach that searches for feature descriptor matches in a persistent database and then applies a Kalman update to all possible matches. Matches are Mahalanobis-distance gated and pruned based on the relative number of observed landmarks with respect to other hypotheses. [90] take an approach using hypotheses on submaps that are scored against the global map based on the innovation they provide to the local odometry trajectory. They use grid based representations and the generation of hypotheses is not within scope of their work. [63] use a nearest neighbour filter and a computer simulation to show global localization without unique landmarks. However, they assume that they can initialize hypotheses by inverting the measurement function to obtain a unique state which is not always possible. A more recent reference work is [49] which focuses on 2D LiDAR global localization using structural unit encoding and multiple hypothesis tracking. The authors mark certain scans from their map as *key scans* and determine endpoint line features. They then form a set of structural units, containing the relations between line endpoints and line angles which is rotation invariant. They then quantize this set using a soft encoding scheme that does not require training on prior data. An exhaustive search is then used to find matching poses from candidate key scans. The authors then use a multi hypothesis

tracking approach where they let the matching poses be the priors, and use the odometry as a likelihood measure to prune unlikely hypotheses from the tree. They compare with the well-known Adaptive Monte Carlo Localization (AMCL, [42], [127]) method and find improved results. However, their method pre-processes the map based on existing scans, while our method does not require existing scan data.

3.2.2 Data association for SLAM

Data association methods are important for Simultaneous Localization and Mapping (SLAM), where they are required for reliable loop closures. One method to deal with finding associations is Random sample Consensus (RANSAC) [41], which randomly selects pairings and evaluates whether the resulting model parameters lead to an acceptable amount of inliers. A large disadvantage is, however, the limited control it provides over the search effort and the fact that it is not guaranteed to find a set of hypotheses in acceptable time. A more structured method is Joint Compatibility Branch and Bound (JCBB) [94], [117]. It searches the so-called *interpretation tree* for the best correspondence between a set of features extracted from sensor data and the map. The method is more reliable than nearest-neighbor association or pair-wise consistency evaluation, as these methods do not take the correlation into account between different observations from the same robot pose. JCBB is shown to be more precise because it evaluates the compatibility of all features seen from a current pose. While the tree structured search of JCBB is similar to our approach, JCBB is not a *hypothesis tracking* approach and as such does not evaluate compatibility over a horizon of motion while maintaining multiple viable pairings. This makes it unsuitable for global localization where different locations appear similar. The use of hypotheses over data associations has also been researched extensively in the context of graph optimization frameworks, see e.g., [81], [20]. In early works such as [128] the authors discuss for a large scale SLAM approach how earlier data associations can be evaluated based on the error they induce on later data associations. They provide a greedy correspondence test to deal with the correspondence problem, but note that other methods may be favourable. Other works such as [28] and [84] focus on methods that either use expectation maximization methods to iteratively search for the data associations or they focus on back-end heuristic methods such as switching variables and robust error models, which allow recovery from false associations. These methods do however generally not provide the multi-hypothesis output on a semantic association level or are only applicable to tracking with a good initialization. Another important work in robust localization that also focuses on existing floor plans is [14], where the authors show how to use these floor plans as priors for localization and maintaining an updated map based on a pose graph framework. However, the authors do not focus on global localization in their work.

3.2.3 Particle filter methods

Particle filters deal with the problem of global localization by covering the location state space with a discrete set of samples (hypotheses) generated from a proposal distribution. They are currently a popular method for representing hypotheses in the pose space (see, e.g., [44], [127], [38]). The combination of particle filters with LiDAR sensors is often considered a *state-of-the-practice* solution to robot navigation in indoor scenarios. Not in the least because easy-to-use reference implementations exist. A recent work on global localization is [6], where the authors add an initialization step to a particle filter that uses indexes generated on the map to find suitable initial positions for the particle filter. They show that this initialization step greatly reduces the initial number of particles needed to let the robot localize itself. The authors pre-compute a set of laser readings in a grid map for over a million positions and use a two-level index to retrieve initial positions and orientations that are likely to match the sensor reading.

3.2.4 Place recognition methods

Some of the methods mentioned such as [6] incorporated ideas from information retrieval. These methods are often used with cameras and reference image databases (see e.g., [7]) and may be used to speed up association initialization but are not exhaustive, and may not work well with existing maps and LiDARs because of lack of unambiguous local context. Furthermore, they often require training on large data sets that are processed to determine a vocabulary, and this is not readily available in every environment. The methods also do typically not include the motion of the robot to gather context or exploit the online recursive nature of the localization process and are therefor significantly different from our proposed method.

3.2.5 Contributions

The contributions of this chapter to the state of the art are the following:

1. We group measurements into local features of type *line*, *circle* and *corner*, resulting in a sparse and accurate local map that is associated with the global map.

2. We assess the lack-of-fit using a horizon of these features that includes the new candidate association, without resolving the robot pose to a belief distribution.

3. We perform an experimental comparison with a commonly used particle filtering method and discuss the trade offs.

In contrast to the works [49], [6], which also perform global localization using 2D LiDAR, we do not rely on grid maps or raw sensor data as a prior and as such do

not leverage data-driven heuristics. We specifically aim to use object-based prior representations that can be obtained from other sources as well (e.g. [59]) in the form of vector maps. This makes our work more similar to [13]; however, the authors do not focus on global localization in their work. Finally, our work draws inspiration from [4], [56] as we consider similar map representations and features. However, their work hypothesizes on individual measurements and [4] attaches information filters to individual hypotheses. Furthermore, [56] uses only line features while we incorporate more general feature types. Our approach reduces the data association space by first associating features locally, and is more in line with how humans reason about localization (e.g., we merge local context from different viewpoints into local features before deciding to relate observations to a map). Furthermore, our framework employs a Bayesian strategy to keep track of hypothesis likelihood. Finally, we compare our work to a often used grid map based particle filter, where the grid map only contains static geometry. The rationale behind this comparison is that grid maps can be easily generated from prior maps as well and provide access to the static geometry without relying on features to be detected, which brings an important trade-off to light. In addition to quantification of our method, we also show conceptual advantages in the form of increased insight in the associations of the method when compared to traditional approaches. Our implementation is available in our repository[1]

3.3 Preliminaries

In this section we introduce preliminary concepts and explain the rationale and assumptions that underlie our method. We approach localization from a *data association* point-of view. Our goal is to have correct associations from *local* (i.e., sensed) feature measurements to features on the global map. In general, these associations are dependent on the global pose, and obtaining one from the other is straight forward. We argue that the benefit of data associations is that they form a finite discrete sample space that can indirectly represent pose distributions efficiently that must otherwise be approximated by samples. It is possible to calculate the belief over global pose from associations (or vice versa) from the general probability distribution:

$$\max_{\mathcal{D}^z, \mathcal{M}, \mathcal{X}^m} P(\mathcal{D}^z, \mathcal{M}, \mathcal{X}^m | \mathcal{Z}). \tag{3.1}$$

Where $\mathcal{D}^z = \{D_1^z \dots D_{n^z}^z\}$ are the data associations of the n_z individual measurements $\mathcal{Z} = \{z_1 \dots z_{n^z}\}$ (for instance, bearing measurements) with the n_m features on the *global map* $\mathcal{M} = \{m_1 \dots m_{n^m}\}$. This general representation allows to solve over the data associations, robot poses in the map frame ($\mathcal{X}^m = \{X_1 \dots X_{n_x}\}$) and map features simultaneously (assuming that the global map is probabilistic). Evaluating (3.1) over both the continuous variables and the full association space

[1] https://gitlab.tue.nl/s135700/semantic_graph_localization

$D_1^z \times \ldots \times D_{n^z}^z$ is generally unnecessary because of conditional independencies. In our method, we first reduce the data association space by forming a local feature map $\mathbf{Y}$, exploiting the fact that measurements can be combined locally into features first and are correlated. Furthermore, we assume that the global map that is available is accurate and that the relative local map coordinates are accurate enough over short horizons of measurements to be independent of the global map, given the measurements. Let $\mathbf{Y}$ be the local map features corresponding to multiple measurements, and $\mathbf{X}$ be the local robot poses. We assume that the *maximimum a-posteriori* (MAP) estimate:

$$\mathbf{X}^*, \mathbf{Y}^* = \arg\max P(\mathbf{X}, \mathbf{Y}|\mathbf{Z}), \tag{3.2}$$

in a local coordinate frame ℓ, can be used to estimate the data associations D^y of the local features with the global map. Here we assume that the data associations between sensor features and features *within* the *local map* can be reliably made. The resulting space of D^y is much smaller than $\mathcal{D}^z$. The remaining problem then becomes to estimate the data associations $\mathcal{D}^y$ given the local map estimate:

$$\max_{D^y, T_\ell^m} P(D^y, T_\ell^m | \mathbf{Y}^*, \mathcal{M}). \tag{3.3}$$

Where the local map estimate horizon is chosen small enough and $\mathbf{Y}^*$ is recomputed for different evaluated patches, such that the assumption of local metric accuracy is met. The pose T_ℓ^m denotes the registration of the local map with the global map, which may be of interest depending on the type of motion control that is used (i.e., *local feature-based motion control* or *global path following*). Because we focus on global localization, we explicitly choose to solve a local factor graph only, and do not introduce the global priors in the estimate. This results in only a single factor graph having to be solved as opposed to multiple. Furthermore, when *global localization* is the current robot task, motion actions and local sensing can (and should) be performed in such a way as to achieve reliable local estimates (e.g., slow driving and conservative detection policies).

3.3.1 Local state estimation

The state of the *local feature map* is represented using primitive features which are estimated together with robot poses using a graph-based optimization, as is common in SLAM literature. We denote the robot pose at time t as $\mathbf{x}_t^\ell = [x_t, y_t, \theta_t]^T$ in a local coordinate frame ℓ. The pose of this frame is arbitrary (usually the starting location of the robot), and is only used for intermediate representation of numeric coordinates. We represent the state of local landmarks by the variables $\mathbf{y}_i^\ell$ which are measured by feature measurements $\mathbf{z}_j$ from the robot poses. If we denote the current optimization horizon of robot poses and landmarks by the sets $\mathbf{X}$ and $\mathbf{Y}$, and the relevant set of measured variables by $\mathbf{Z}$, we can write the maximum a-posteriori estimate of the these variables given the measurements as:

$$P(\mathbf{X}, \mathbf{Y}|\mathbf{Z}) \propto P(\mathbf{Z}|\mathbf{X}, \mathbf{Y}) P(\mathbf{X}) P(\mathbf{Y}) \tag{3.4}$$

Where we have assumed independence of measurements. The measurement likelihood $P(\mathbf{Z}|\mathbf{X},\mathbf{Y})$ in the above equation is encoded by *factors* in a *factor graph*. These factors take the form of nonlinear measurement functions $\mathbf{h}_j$ of the state where the uncertainty is represented by additive zero-mean multivariate Gaussian error, taking the form:

$$P(\mathbf{z_j}|\mathbf{X}_j,\mathbf{Y}_j) \propto \exp\left(-\frac{1}{2} \parallel \mathbf{h}_j(\mathbf{X}_j,\mathbf{Y}_j) - \mathbf{z_j} \parallel^2_{\Sigma_j}\right) \tag{3.5}$$

Where $\parallel \cdot \parallel_{\Sigma_j}$ denotes the Mahalanobis distance, using the measurement covariance matrix Σ_j. A nonlinear least-squares problem is obtained by taking the log of the product of factors, leading to the formulation:

$$\log P(\mathbf{X},\mathbf{Y}|\mathbf{Z}) \propto -\sum_{j=0}^{K} \mathbf{e_j}^T \Sigma_j^{-1} \mathbf{e_j} \tag{3.6}$$

where we use $\mathbf{e}_j = \mathbf{h}_j(\mathbf{X}_j,\mathbf{Y}_j) - \mathbf{z_j}$ for all K measurements. The resulting problem is a minimization over the negative log likelihood, resulting in the non-linear least squares formulation:

$$\min_{\mathbf{X},\mathbf{Y}} \sum_{j=0}^{K} \mathbf{e_j}^T \Sigma_j^{-1} \mathbf{e_j} \tag{3.7}$$

where we assumed uninformative priors, leaving only factors in the optimization. The resulting maximum a-posteriori (MAP) estimate contains the most likely local map and robot poses given the measurements. Here we assume that all data associations between local features and their measurements are correct and can be reliably made. It is often possible to design features and feature detectors with enough saliency and discrimination such that this assumption holds locally, given proprioceptive sensing that is relatively accurate over short distances. We use the GTSAM [32] library to perform the optimization.

3.3.2 Geometric primitives

We use the term *measurements* to describe the primitive *sensor features* that are extracted from raw sensor data and that are associated with features on the *local map*. We refer to the latter as *local map features* which can be associated with multiple measurements, and their geometry must be consistently inferred from the *sensor features* belonging to the measurements. An example of a local map feature can be a circle, which is associated with multiple circle sensor features from different LiDAR scans. measurements also contain *spatial* information about the location of the features with respect to the robot. these are in the form of *range / bearing / pose* measurements. In this work we use three *geometric primitives* that form the basis for landmarks in our localization approach; *line segments*, *corners* and *circles*. Note that these features resemble three general geometric constraints on the state of the robot given the measurement, by reducing the likely state to

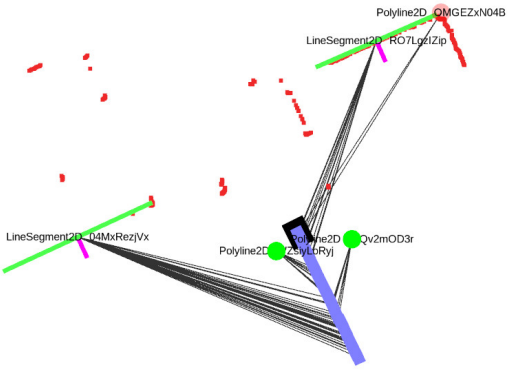


Figure 3.2. *A small locally consistent map of line and corner features, that has been obtained by locally associating and merging feature representations and solving the factor graph optimization problem after each new LiDAR scan is obtained. The robot poses are shown in blue with the current pose in black. measurement associations are shown as grey lines and current LiDAR points in red. The line measurements include a normal to indicate their visible side. A corner detection is shown in translucent red when not enough measurements have been associated with it (candidate detection). The features get assigned unique string identifiers.*

either a line segment, a pose or a circle around a point measurement. These feature types can therefore be generalized to other sensors as well. A line segment is parametrized by two points which represent the *begin* and *end* of the line segment. A *partial* line segment is any combination of two points that are contained within a line segment, as is often the case for LiDAR measurements. A corner is modeled by a *polyline* consisting of three points, i.e., two line segments sharing a common point. A *partial* corner is again any corner where the outer points lie *within* the actual line segments. Note that a LiDAR allows to detect approximate end-points of lines, by evaluating whether the neighboring range extends beyond the line segment or is visually obscured by something closer. Finally, a circle is modeled by a center point and a radius. An example of a local map created from sensor data can be seen in Figure 3.2. We will now proceed while assuming that such a local map is available. A more detailed discussion on the mechanisms used to create the local map has been deferred to Appendix A.

3.4 Localization Approach

In this section we will explain our global localization method. We will first elaborate on how we maintain our local map in relation to our global objective and then introduce the global association approach using a hypothesis tree.

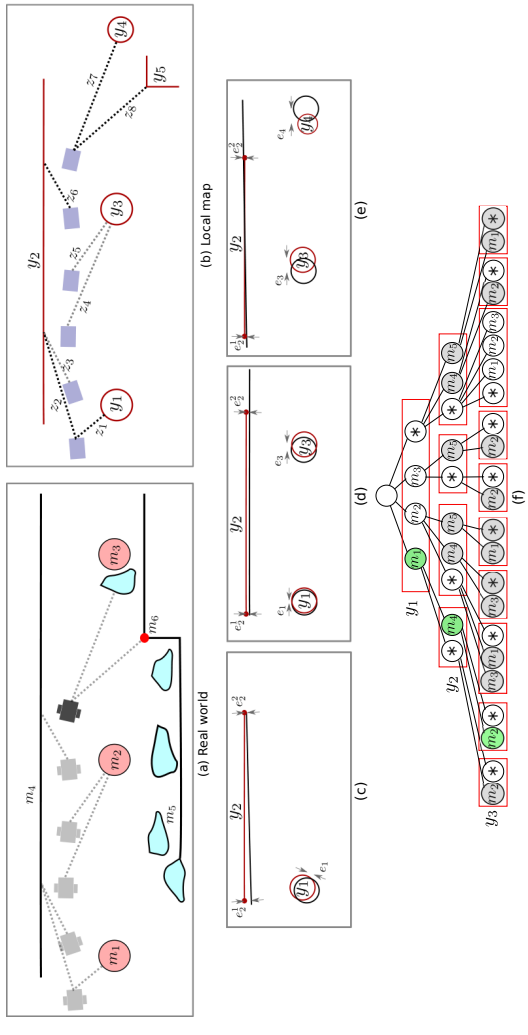


Figure 3.3. *A basic example showing the global association tree. As the robot drives in the real world (a), it builds a local map of features (b). The local measurements are associated with these local features, and added to their measurement set. Nodes are then used to represent an association between a local feature and a feature on the global map in (f). In this example, the robot first spots a circular feature. After spotting it multiple times, the consecutive measurements are added to a measurement set and a local feature is instantiated. Possible pairing hypotheses with the global map are generated based on feature type and dimensions. When a new line segment is detected, a second layer of hypotheses is added to the tree based on spatial consistency between the circle and line feature on the local map and the global map. Finally, an* unmapped object *is spotted which has similar geometry to a column, highlighting the importance of allowing for a* unmapped object *option in the association tree, although these levels are not visualized anymore. In sub figure (c-e) the spatial congruency is visualized, which is used together with the feature's descriptive component to accept or reject association hypotheses.*

3.4.1 Local mapping

The map that we maintain locally is updated for each new sensor feature. Once a new sensor feature is extracted, the map is updated using the approach described in algorithm 1. The sensor feature extraction is triggered by a distance monitor

Algorithm 1 The algorithm used to update the local map features when a new set of sensor features has been extracted from a laser scan.

Input: sens. feats $f_1 \dots f_N$, factor graph L, odom T_{od}
update_local_map(f,L, T_{od}):
robot pose $x_t^\ell = T_{\text{od}} x_{t-1}^\ell$
L $\leftarrow$ L $\cup$ T_{od}, x_t^ℓ
for f_i in $f_1 \dots f_N$ **do**
  associated = false
  f_i^ℓ = transform_from_sensor(f_i, x_t^ℓ)
  for y_i in L.features **do**
    if eq_type(f_i,y_i) && eq_dim(f_i,y_i) && within_thres(f_i^ℓ,y_i) **then**
      z_i = range_bearing(f_i)
      L.update(f_i,y_i,z_i)
      $\mathcal{G}$ =set_of(y_i)
      $\mathcal{G}$ $\leftarrow$ $\mathcal{G} \cup z_i$
      associated = true
    end if
  end for
  if associated == false **then**
    y = create_local(f_i^ℓ)
    z_i = range_bearing(f_i)
    L $\leftarrow$ L $\cup$ y, z_i
    $\mathcal{G}$ =create_set(y_i, z_i)
  end if
end for
L.coordinates $\leftarrow$ least_squares(L)
return L

based on wheel odometry and the latest available sensor reading is combined with a time-stamp interpolated version of the odometry encoder reading. The updating routine (algorithm 1) shows how a sensor feature is first checked for existing associations with the local map. Here, the functions `eq_type()` and `eq_dim()` check if the feature has the right type and dimension (e.g.. a circle with similar radius). The function `within_thres()` checks if the feature is close enough to be associated, where we use a Euclidean distance. If no such association exists, a new local map feature is instantiated. For this purpose, we introduce the notion of a measurement set $\mathcal{G}_i$ which contains a local map feature and all measurements associated with it[2].

[2]While a measurement set is technically a redundant notion, since measurements are already associated with a single feature, it makes bookkeeping and implementation more clear

3.4.2 Global association tree

We maintain hypotheses over the space of data associations between the local and global map. These hypotheses will be maintained based on explicitly configurable assumptions, while a Bayesian approach is used to update scores for hypotheses. Consider a snapshot of a given local map with a feature $y_i \in \mathbf{Y}$. We define the association variable D_i as follows:

$$D_i = \begin{cases} d_i^1, & y_i \text{ originates (partly) from } m_1 \\ \dots & \\ d_i^n, & y_i \text{ originates (partly) from } m_n \\ d_i^*, & y_i \text{ originates from } \textit{unmapped object} \end{cases} \tag{3.8}$$

Here, unmapped object refers to something that is not on the *global* map. We explicitly mention *partly* because we allow cases where the global feature m_j is responsible for only part of the geometry of y_i, as long as the effective constraints is the same (e.g. a wall that appears extended by an object from a certain viewpoint). We will now consider how the association variables form the association tree that we will refer to in the remainder of this work. Having a local map available, we evaluate (3.3) by starting with the first feature and expanding the tree for each consecutive feature. The *levels* of the association tree coincide with the local map feature indices, i.e., nodes h_{ij} on level i pair y_i with a global map feature. The leaf nodes of the tree represent an association assignment of all local map features up to y_i, obtained by following the parents par(h_{ij}) of the nodes upwards. The leaf nodes thereby form the hypotheses under consideration. Each node thus represents the association between a local feature (and thereby its measurement set) and a global feature, while pointing to a parent node that it is conditioned on[3]. Nodes on a single level i can associate a local map feature y_i with the same global map features, if their parents are different. We can require a measurement set to meet certain constraints on the measurements and local features before an association is made or revisited. For example, we can require a certain minimal number of measurements from different poses. The structure that we describe naturally forms a hypothesis tree $\mathcal{T}$ and the current *sample space* is formed by the *leaf nodes*, which all represent a unique set of data associations between local map features and global map features. Note that because of the inclusion of *unmapped object* associations, this sample space is *exhaustive* when no pruning has been applied. An example is provided in Figure 3.3. As the robot drives, it detects a circular feature, and a measurement set $\mathcal{G}_1$ is created. Based on the shape and size of the feature, a pairing can be made with possible features on the map. Alternatively, it can be something that is not on the map $(*)$. When a next line segment feature is spotted, a new set $\mathcal{G}_2$ is formed and new leaf hypotheses are formed for each old leaf hypothesis. Based on the old hypothesis, we will now consider when associations are feasible and when they are not. Feature shape is used first to select

[3]Except for the root hypothesis, which only serves as a placeholder for the first layer of children, and may hold optional prior information supplied by the user regarding location

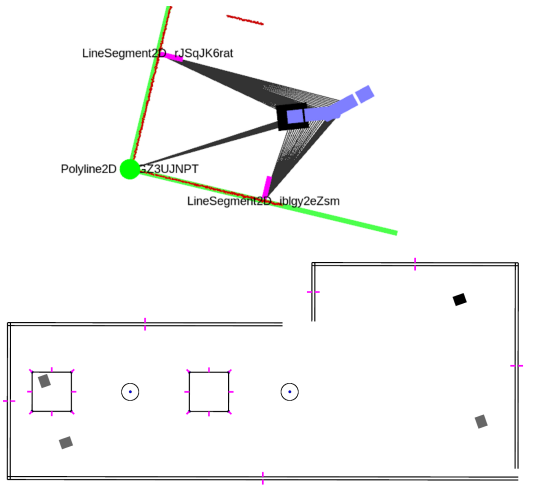


Figure 3.4. *The global map hypotheses (bottom) corresponding to a local map (top). Based on the local features in that figure, four possible map poses are shown (not all hypotheses permit showing a unique pose). Note that walls have been modeled by double line segments with visibility normals visualized by small purple segments. The labels of the global map have been omitted in this example.*

plausible candidates, after which we assess whether the *spatial structure* for a small horizon of recent features is congruent with the spatial structure obtained from the map (Fig. 3.3c). This amounts to assessing the *spatial likelihood* of the local map (Fig. 3.3b) given the data associations. Depending on the specific representation of the map, features and uncertainty, the likelihood can take many forms (e.g. topological, semantic or spatial). But we choose to do a spatial registration step and assess whether the resulting spatial mismatch is small enough to refrain from rejecting the pairing hypothesis. At this stage, making as little feasible hypotheses as possible and pruning unlikely leaves becomes necessary to avoid a combinatorial explosion. For hypotheses marked in grey, a unique maximum likelihood pose on the map can be determined based on the feature pairings. However, for assessing the geometric *lack-of-fit* of local detections, such a pose is in general not necessary (e.g. in the case of parallel line segments).

3.4.3 Hypothesis likelihood

Having a window of local map features available, the problem under consideration is to determine which association hypotheses are likely and which are not. The main mechanism to keep a small set of hypotheses is to prune unlikely associations immediately from the tree. Scoring hypothesis likelihood recursively is not a strict necessity to achieve this in our method, as we rather use more direct approaches for rejection. However, maintaining likelihood is useful for three reasons: 1) it

allows to keep the most likely hypothesis when hypotheses are similar (e.g. they share recent associations), 2) it allows to prune the least likely hypotheses when the number grows too large to maintain efficiently and 3) it would allow a planner to optimize actions taking relative uncertainty into account. Regarding the second remark, we note that our method guarantees a qualitative distinction between hypotheses based on their data association, making rigorous pruning possible while still keeping enough descriptive power in the remaining set.

In general, localization does not reduce to merely a *lack-of-fit* evaluation problem, because the likelihood of the associations we make with local map features (including the *unmapped object* pairing) is also dependent on the *local structure* of the feature (e.g. shape, size, semantics) which is independent of position. Furthermore, the nature of disturbances in the real world and the feature density on the map play a role in determining prior likelihood for a global association. We now explain our Bayesian strategy for maintaining a *breadth-first* expansion of the tree, allowing us to express confidence using models of both spatial and descriptive likelihood. We employ a *gating* or *constraint* procedure based on feature description and spatial error, leaving us with only a certain subset of hypotheses that should be considered likely. In general, we can evaluate the association probabilities using the product rule:

$$P(D|\mathbf{Y}^*, \mathcal{M}) = \prod_{i=1}^{n^y} P(D_i|D_{i-1:1}, \mathbf{Y}^*, \mathcal{M}). \tag{3.9}$$

followed by Bayes rule to evaluate in terms of spatial likelihood and prior:

$$P(D_i|D_{1:i-1}, \mathbf{Y}^*, \mathcal{M}) \propto P(\mathbf{Y}^*|D_{1:i}, \mathcal{M})P(D_i|D_{1:i-1}, \mathcal{M}) \tag{3.10}$$

Where the spatial likelihood allows us to evaluate whether the local map that we observe is coherent with the global map given the data associations. In other words, once we evaluate a new feature pairing D_i for a local map feature y_i, we evaluate the spatial likelihood $P(\mathbf{Y}^*|D_{1:i}, \mathcal{M})$ of the local map, given the feature pairing and its parent pairings up to a horizon N_d resulting in a recursive evaluation. We can also incorporate a prior $P(D_i|D_{1:i-1}, \mathcal{M})$ for this pairing, expressing confidence based on feature description (e.g. semantics, shape, size). In practice, this prior will be assumed independent of prior associations $D_{1:i-1}$. The resulting hypothesis likelihood is then obtained by multiplying with the likelihood of the parent node as expressed in (3.9). We evaluate the spatial likelihood by taking a small set N_d of recently added local features that together are enough to determine a pose registration *including* the current feature y_i and data association D_i. This pose registration is obtained by minimizing the squared error between the local and global map on that horizon:

$$L^* = \min \frac{1}{N_d} \sum_{k=1}^{N_d} w_k e_k^2, \tag{3.11}$$

where the error term is either the Euclidean or the projected distance:

$$e_k^2 = \begin{cases} \sum_{s\in\{1,2\}} \parallel n_k \cdot (m_k - (Ry_k^s + t)) \parallel^2, & \text{for lines} \\ \parallel (m_k - (Ry_k + t)) \parallel^2, & \text{for points,} \end{cases} \tag{3.12}$$

With R and t the rotation matrix and translation vector that together form the registration pose. The upper expression in (3.12) is used for matching line segments to lines using the normal n_k for both line endpoint y_k^1 and y_k^2. The lower expression is used for matching regular point positions. The feature set used in (3.11) includes the current candidate feature, allowing us to evaluate the *lack-of-fit* with the new feature influencing the registration. The errors are illustrated in Figure 3.3. We treat corner points as points in this registration and disregard their orientation in obtaining the registration. The unnormalized likelihood is now determined by considering the error distance of the points, either Euclidean or projected, after aligning them using the obtained registration:

$$p(\mathbf{e}) = \prod_{i=1}^{N_d} \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left\{-\frac{e_i^2}{2\sigma_i^2}\right\} \tag{3.13}$$

$$\ln \mathcal{L}(\mathbf{Y}^*; D_{i:1}, \mathcal{M}) \approx -\frac{1}{N_d} \sum_{i=1}^{N_d} \frac{e_i^2}{\sigma_i^2} \tag{3.14}$$

where σ_i is the error covariance of feature i on the local map.

3.4.4 Implementation

We use the well-known singular value decomposition (SVD) approach to obtain the solution to (3.11), since correspondences are already provided by the hypothesis under consideration. Figure 3.5 schematically shows how we use a pre-alignment step based on line intersection points to first determine an approximate registration. This allows to then project the line segment endpoints onto the infinite line extensions on the global map and incorporate them into the registration as points, resulting in an iterative solution. We employ a very coarse gating procedure for hypotheses based on the global pose to avoid pre-aligning and registering features unnecessarily.

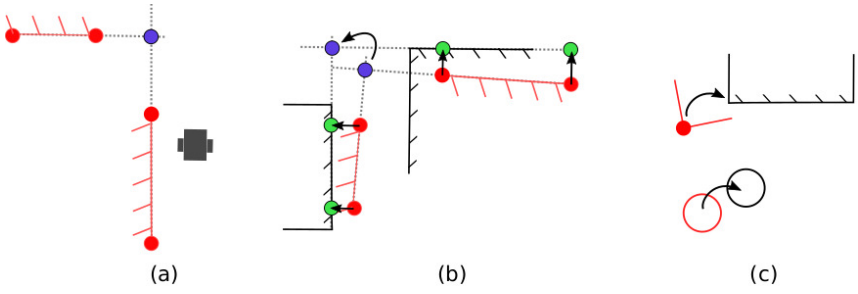


Figure 3.5. *(a) Two (partial) line segments in the local map. (b) A pre-alignment step based on the virtual intersection point (blue) allows to acquire an initial estimate. We also use two point-correspondences or a single corner to obtain this pre-alignment, depending on which is most recently available. The segment endpoints are projected onto the global map line segments (green) to include them in the registration. The blue point is not used because it is only measured by extension. (c) Corners are treated as points in registration, similar to circles.*

For the weights in (3.12), a covariance estimate can be used from the local mapping. However, we supply simply unit weights because the covariance matrix obtained from the local map is, unlike the maximum a posteriori estimate, not necessarily an accurate estimate due to linearization and assumption of measurement independence. The method we propose should not be overly sensitive to these kind of modeling decisions. Some special attention has to be paid to the instances where the features do not permit a unique registration. In these cases, we evaluate a similar error metric for the *feature distance* that can be obtained without considering a registration. We distinguish between the following cases:

- The initial feature; in this case we only consider the descriptive prior likelihood for the data association.
- Parallel line segments; in this case we evaluate the distance between the segments.
- The *unmapped object* option; We use a heuristic probability that is small, but data driven estimates may provide better results that reflect sensor and environment characteristics.
- Any other case; where we can evaluate the likelihood as explained previously.

3.4.5 Pruning

We will now introduce the mechanisms we employ to remove unlikely hypotheses from the tree. When considering a new local feature, first a gating step checks for every global map candidate whether the shape and size of the feature correspond and - if a pose estimate is available - whether the feature is roughly at the correct location. The latter gate is to prevent unnecessary evaluation during the spatial

congruency check. For the congruency check, a small horizon of features including the new association candidate is registered according to criterium (3.12) and checked for errors that exceed a spatial threshold, as shown in Figure 3.6.

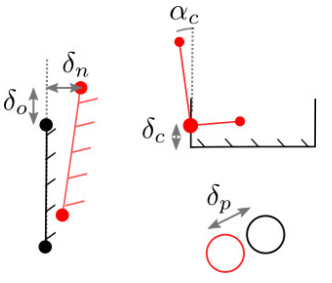


Figure 3.6. *The thresholds that are checked to decide if a hypothesis should be rejected.*

If any of the features exceeds this threshold the pairing hypothesis is rejected. Early pruning will limit the tree growth and prevent combinatorial explosion. However, it will not stop an existing hypothesis from expanding, even if there are no new local features supporting it. This is due to the explicit *not-on-map* option that is always expanded to allow delayed evidence[4]. To this end, we remove *redundant* hypotheses by checking whether hypotheses make similar associations in their history (excluding not-on-map) and keeping only the maximum likelihood hypothesis among them. Formally, for every leaf node (i.e., hypothesis), we recursively evaluate the parent nodes up to depth N_{sim} and consider leaf nodes similar if and only if they do not contain $*$-associations and all associations are the same. We then only keep the leaf node that has the highest likelihood. We also limit the number of consecutive not-on-map associations allowed for any hypothesis to γ_* to prevent unbounded tree growth. This parameter balances recovery potential with computational effort. Finally we have a pruning step that reduces the number of hypotheses to a fixed maximum N_{max}, either based on likelihood or amount of *not-on-map* associations. The latter keeps *at least* N_p hypotheses, by choosing the smallest n_* where any hypothesis with more then n_* *not-on-map* associations is removed. This procedure is applied after convergence of the hypotheses to leave only a small set of *back-up* hypotheses remaining to recover locally from a false association.

3.4.6 Implementation considerations

In our work we do not explicitly represent probability distributions over a global pose. This sets our method apart from pose-based filtering methods. However, determining a maximum likelihood pose using our methods is straight forward

[4]Note that we never refrain from making a *not-on-map* option, because erroneous pairings can appear confident due to error. The delayed-evidence mechanism is the major reason for robustness against this

Algorithm 2 The algorithm used to evaluate the hypotheses once a new set of local map features is available. It assumes that the tree already contains one or more levels. If the tree is empty, only feature shape is considered for the first local map feature.

Input: loc. map feat. $y_1 \dots y_N$, glob. map $\mathcal{M}$ hyp. tree $\mathcal{T}$

```
evaluate_hypotheses(y,M,T):
for y in y_1 ... y_N do
  for h in T.get_leaves() do
    if h.registration_cached() then
      T_ℓ^m = h.registration()
      y^m = T_ℓ^m y
    end if
    for m in M do
      if shape_eq(y,m) && spat_gate(y^m,m) then
        M = get_horizon(M, T,h,N_d)
        Y = get_horizon(Y, T,h,N_d)
        Y_pre = pre-align(M,Y)
        e,T_ℓ^m,l = regist_least_sq(M,Y_pre)
        if ¬ eval_for_outliers(e) then
          h_new = make_hyp(y,m,h,T_ℓ^m)
          h_new.log_l = h.log_l + l
          T.add_hypothesis(h_new)
        end if
      end if
    end for
  end for
  T.add_clutter_hyp(y,h)
end for
T.prune_similar_hypotheses(N_sim)
T.prune_consec_*(γ_*)
T.keep_max_likelihood(N_max)
return T
```

given the data associations, and the registration obtained by (3.11) can be used for this purpose. A consequence is, furthermore, that we have access to a locally accurate, association-labelled map which may be used by motion planners. This is valuable, for instance, in a hallway where two parallel sides are annotated and permit a driving actions without relating to a pose. A consequence is that we need to determine the length of the local horizon we consider. The local map is chosen sufficiently large to enable considering a horizon of features for global association, but should not be seen as a means of closing *large loops*, i.e., loops that require a substantial drift to be corrected. Because this process is potentially error-prone and unnecessary for our localization objective.

Finally, if the objective under consideration changes to metrically accurate tracking, it may be favourable to replace our *breadth-first* expansion approach to *depth-first* tree expansion. In that case, one can provide the local graph with *global metric priors* obtained from the associations to obtain a more accurate estimate of the map position under the assumption that the map is of reasonable metric accuracy in the first place. backtracking in this depth-first approach based on feature monitoring to recover quickly from false associations is a topic for future work.

3.5 Experimental Evaluation

In this section, we evaluate our approach in an indoor environment for which a map is available. We use this map to localize a teleoperated robot and we compare our method to the well-known AMCL [42] particle filter implementation. AMCL is openly available and uses a different paradigm for localization based on grid maps and beam-hit sensor models, which can be applied to global localization. We will validate and compare our method using criteria for success ratio, convergence speed and metric accuracy for different starting locations and different degrees of disturbance and prior knowledge. The real world experiments allow us to assess the performance qualitatively in scenarios with varying levels of uncertainty that we can add or remove in the form of prior knowledge and clutter. We use a custom-made platform equipped with mecanum wheels, wheel encoder odometry and a Hokuyo UTM30-LX 2D LiDAR scanner mounted upside down, close to the floor (Figure 3.8). Due to its mounting position, we have a 180 ° field of view consisting of 720 scan points. We choose this sensor type for evaluation because it is often encountered on mid-range existing robot platforms in real world applications.

3.5.1 Indoor environment

We use a part of the Atlas building on the campus of the Eindhoven University of Technology for the experimental evaluation. Items that are moved often such as chairs, tables, carts and dustbins are not represented on the map that we use (sized approx. 20 x 41 meters) and form the *unmapped object* that our method should be robust against. Furthermore, in part of our data set, bystanders disturb the view

of the robot and we place objects in the scene that either obscure or resemble the static features that our method uses, which are likely scenarios. The map of the environment together with the initial starting positions, and a sub-imposed grid map is shown in Figure 3.9. Note that this is a SLAM-generated map that is only used for illustration. Based on the static vector geometry in Figure 3.9, we will generate a new grid map containing only the static parts used for our comparison. In Figure 3.8 we show the environment, some added disturbances for the final scenario and the robot platform.

3.5.2 Evaluated scenarios

We consider the following scenarios labeled A-C:

A. This scenario considers two initial positions in the map, that are all within a starting region (sized approx. 15 x 15 meters) known to the robot, shown in Figure 3.9.

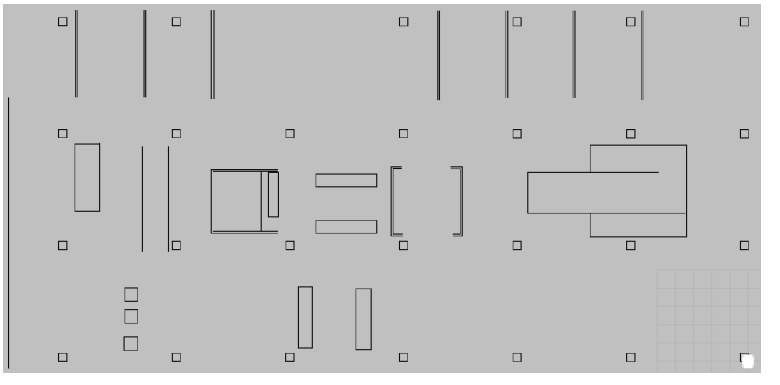
B. This scenario considers ten different initial positions all over the map, where the starting region is unknown to the robot, containing only encountered disturbances (including people).

C. This scenario considers added disturbances to the environment, in the form of deliberately obscured features.

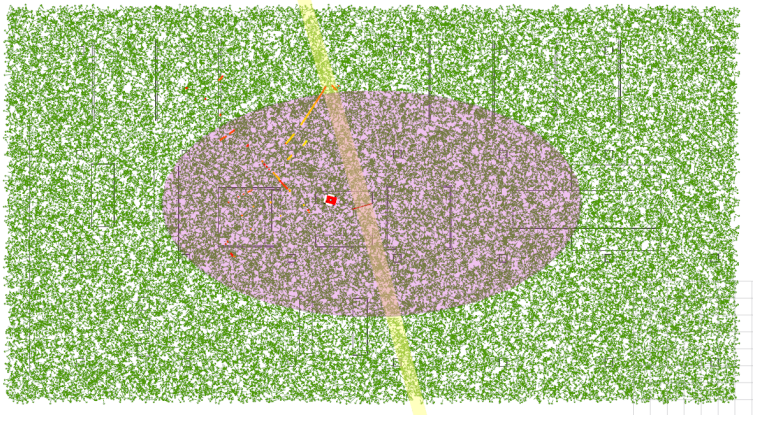
The scenarios are evaluated for a total of 12 different starting positions. The final scenario (C) is evaluated for two initial positions, leading to a total of 14 evaluated global localization challenges. Our localization method exploits the initial knowledge in scenario (A) by transforming the starting position of the robot in the local map to the global map and rejecting hypotheses that were not started within the indicated region. For AMCL, the initial sample set is confined to the region.

3.5.3 Comparing with monte-carlo localization

We compare the performance of our global localization approach with the monte carlo localization (MCL) approach based on grid maps. The reason is that in this localization paradigm, the robot does not rely on feature extraction and local mapping, due to the grid cell decomposition and pose sampling approach. In contrast to our proposed method, AMCL uses random (re-)sampling of hypotheses. We will run AMCL five times for every trajectory to account for this randomness (leading to a total of 70 AMCL runs). We choose the ROS AMCL implementation as it is a very commonly used software package using an adaptive sampling scheme. For this comparison, we generate the grid map for AMCL based on the vector representation of the static geometry in the environment. This vector map was made manually by referencing the actual environment and only includes objects on the map that do not move (i.e., no moveable furniture, tables, carts, trash bins).



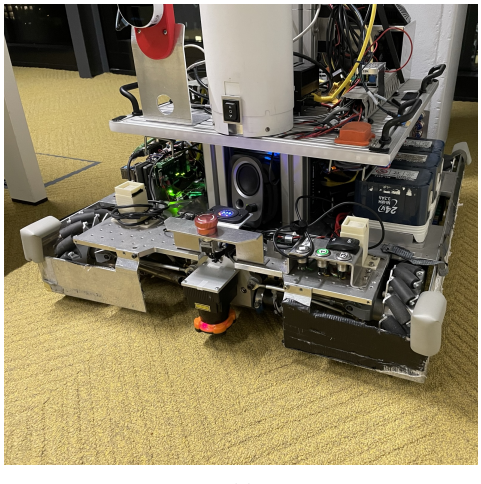
(a)



(b)

Figure 3.7. *(a) The grid map containing the static geometry as generated for AMCL localization. (b) The initial particle spread in AMCL, consisting of 50,000 particles.*

(a) (b)



(c)

Figure 3.8. *(a) The robot platform in the Atlas environment, the position is indicated in the map in Figure 3.9 position 1 (oriented towards the left). The situation shown is used as scenario A, where unmapped items are on the map in the form of chairs, tables and carts. (b) Some of the disturbances added to a part of the map to purposely obscure or mimick features for scenario C. (c) A close up of the robot platform with the planar LiDAR sensor.*

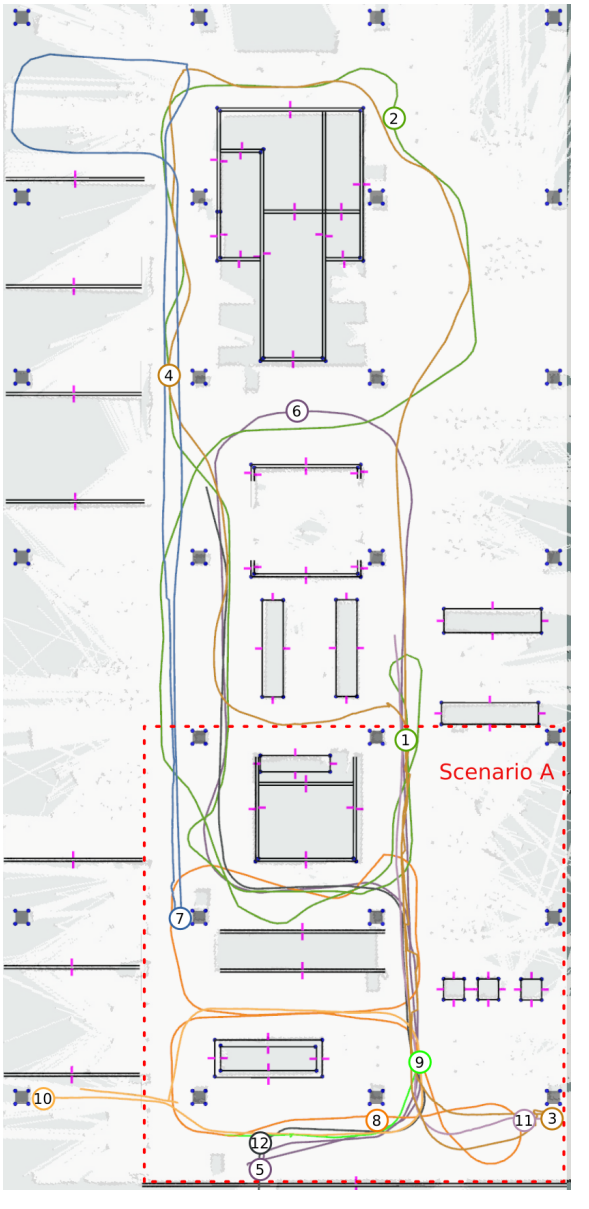


Figure 3.9. *A map of the environment, with the trajectories driven by the robot indicated as paths and their starting locations (in different colors for clarity). The map contains line segments, corners (blue points) and visible normals indicated in purple. The world contains unmapped objects that are encountered during every day use of the environment. To show how these unmapped items influence the LiDAR readings, an occupancy grid map has been aligned to the map for visualization purposes only. The features shown are linked to semantic object instances, but these are omitted in the figure for clarity. The starting region for scenario A is shown in red.*

Table 3.1. *The configuration parameters used in our method for evaluating the hypothesis tree.*

param.	value	descr.
N_{max}	200	max. # hypotheses
N_{sim}	2	similar associations before prune
γ_*	3	max. consecutive not-on-map assoc.
σ_e	1	error covariance
$p(*)$	0.1	probability not-on-map assoc.
$\delta_o, \delta_n, \delta_c, \delta_p$	0.5	error tolerance before prune
α_c	0.5	error angle tolerance before prune

The grid map is shown in Figure 3.7 and was generated automatically from the vector representation in Figure 3.9, by setting grid cells to occupied if they cross the vector geometry. This way, both our approach and AMCL rely on a representation of the environment that can be obtained from a sensor-independent source that does not contain all the move-able geometry (*unmapped objects*) in the environment.

3.5.4 Configuration of the approaches

In our feature-based approach, we select a minimal length of 0.2 meters for corners and 1.5 meters for line segments before taking them into account as features. The latter is heuristically chosen to avoid the detection of open doors as line segments. No circle shaped features are present in this environment, so they are not used. We trigger a perception update after every 0.05 meters of driving and 0.05 radians of rotation (which in practice is higher because of loop time constraints, which we will consider later). We only trigger association tree evaluation when new features are available on the local map with at least three measurement from different poses. The tree evaluation parameters are given in Table 3.1. We show the grid map that was generated for AMCL in Figure 3.7, with a discretization of 0.05 m. For the AMCL configuration, the parameters that we used are in Table 3.2. We choose the likelihood field sensor model over the beam-based model because it showed better results by allowing to simulate more particles. We increase the initial particle count to 50,000, which is found to be the maximum reasonable amount of particles that our system is able to process. The particles are spread evenly over the map by uniform sampling.

3.5.5 Performance metrics

We will run the algorithms real time on a laptop with an Intel Quadcore i7-7700HQ CPU @ 2.80GHz $\times$ 8 processor. We evaluate the following performance metrics:

- Succes: We define success as the maximum likelihood hypothesis reaching and maintaining an error of less than 1.0m within the 60 seconds time frame of the run.
- ML dist: We define the maximum-likelihood (ML) distance as the distance driven in which the previous succes criterium is met.

Table 3.2. *The configuration parameters used for running AMCL*

param.	value	param.	value
$N_{\text{min\_particles}}$	100	$N_{\text{laser\_z\_short}}$	0.1
$N_{\text{max\_particles}}$	50,000	$N_{\text{laser\_z\_max}}$	0.05
$N_{\text{kld\_err}}$	0.01	$N_{\text{laser\_z\_rand}}$	0.05
$N_{\text{update\_min\_d}}$	0.10	$N_{\text{laser\_sigma\_hit}}$	0.2
$N_{\text{update\_min\_a}}$	0.20	$N_{\text{laser\_lambda\_short}}$	0.1
$N_{\text{resample\_interval}}$	2	$N_{\text{laser\_likelihood\_max\_dist}}$	2.0
$N_{\text{transform\_tolerance}}$	0.1	$N_{\text{laser\_model\_type}}$	likelihood_field
$N_{\text{recovery\_alpha\_slow}}$	0.0	$N_{\text{odom\_model\_type}}$	omni
$N_{\text{recovery\_alpha\_fast}}$	0.0	$N_{\text{odom\_alpha1}}$	0.2
$N_{\text{laser\_max\_range}}$	LiDAR value (30)	$N_{\text{odom\_alpha2}}$	0.2
$N_{\text{laser\_max\_beams}}$	30	$N_{\text{odom\_alpha3}}$	0.2
$N_{\text{laser\_z\_hit}}$	0.95	$N_{\text{odom\_alpha4}}$	0.2

- self-rep. distance: We define self-reported distance as the distance in which our method reports no hypothesis that exceeds the maximum deviation of 1.0 m from the weighted average of all hypotheses that permit a pose location. For AMCL, we use a threshold of 2 meters on the x and y pose covariance.

- ML error: We report the error after the particles have converged to the right location, to indicate error in tracking mode.

- Max Hyp: We report the maximum number of hypotheses before and after the particles have converged to the right location.

For AMCL, the indicated values are the average over all successful runs for that initial positions. We will also show some selected results regarding semantic associations and CPU usage for our approach.

3.6 Results and Discussion

The results of the localization are shown in Table 3.3. For the smaller initial region, our method achieves localization within 1.7 and 4.9 meters for A1 and A3 with a maximum number of 200 hypotheses, which reduces to only resp. 8 and 7 hypotheses after self-reported localization. AMCL also achieves localization successfully when the initial pose is confined to the smaller region. When we increase the possible initial location to the entire map (scenario B1-B10), we see that our method still achieves localization in 9/10 runs, with distances ranging from only 1.1 meters for B5 to 10.3 meters for B10. The output of our method before and after localization is shown in Figure 3.12. The symmetries occurring from the feature associations are evident. The pattern shown here consisting of two line segments with visualized normals is already a very strong indicator of location. After localization, the figure shows the back-up hypotheses that enable to recover from wrong associations during tracking. Self-reported distances are usually significantly longer because alternative hypotheses still exist that offer

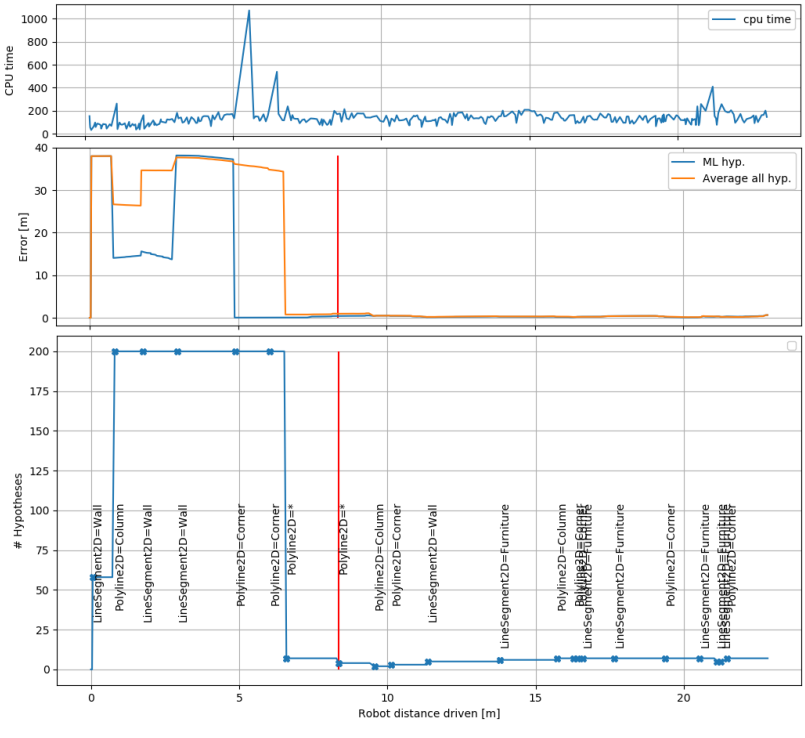


Figure 3.10. *The CPU time, error and hypotheses shown for B3. The middle figure shows the error for both the maximum-likelihood hypotheses and the average over all hypotheses that permit a pose registration. The moment where our approach marks itself as localized is shown in red. Below, the number of hypotheses is shown, together with events that indicate that a new association is evaluated. The association for the ML hypothesis is shown in text (* indicates unmapped object). The top figure shows CPU cycle time (milliseconds) for session B3. Within a cycle, the local map is always updated and the associations are only updated if a new stable local feature is available. The latter causes the distinct peaks in processing time.*

alternative explanations of the local map. In some cases, such as B9, the self-reported distance is too low. This is due to only pose-enabling hypotheses being evaluated for convergence, and many hypotheses that do not permit a pose are present and appear to be correct in light of later evidence. For AMCL, we see that for some runs, such as B3, B8 and B9, successful localization is consistently achieved. In the case of B9, AMCL is very fast to converge to the right location. In B9, as is difficult to see in Figure 3.9, the robot rotates in place for a full round first, then moves to a second location and rotates again, exposing significant geometry of the environment without many disturbing elements present in the environment. Our method takes longer to localize in this scenario, because we rely on detecting stable features first from the geometry. In scenario B7 our method is not able to localize whereas AMCL is able to do so in 3/5 runs. Our method, in this scene, had trouble with the people and furniture in the open space to the left of the robot, which caused drift in our local map. In run B2, B4, B5 and B6, we see that AMCL is consistently not able to localize the robot. Especially in the right half of the map (as can be seen in Figure 3.9), there is clutter present in the B-scenarios. While our method does localize, it results in erroneous hypotheses taking longer to be removed. Finally, in the C-level scenarios we see that AMCL is able to localize successfully (5/5 runs) in scenario C12 and sporadically in scenario C11. For our method, C12 is problematic because most of the objects mapped locally are actually clutter objects that have been placed to disturb the environment (the dustbin and the yellow board in that figure, Figure 3.8)b also shows the divider that was placed). The middle and right image also show scenarios in which clutter appears in the local map that has to be dealt with (i.e, a dustbin and a open door appearing as a corner).

3.7 Associations and Computing Time

Figure 3.10 shows the number of hypotheses and error in time and illustrates an attractive property of our approach. We maintain a smaller number of hypotheses on the basis of symbolic associations, especially after convergence. Furthermore we make the associations very explicit. The new associations made by the ML hypothesis are indicated in the graph as events. The hypothesis tree is only evaluated at these instances for the new evidence. The resulting CPU cycle time is shown in Figure 3.10. The distinct peaks are caused by evaluation of the tree, whereas the local map update is usually between 100 and 200 ms. Note however that our current C++ implementation was written as a prototype with extendability in mind and has not been optimized for execution time. We expect that computation times of less than half the current times are obtainable.

3.7.1 Limitations

The amount of hypotheses needed was determined by considering that we have 154 concave corners on our map. If the first detection would be a corner, we can

Table 3.3. *Results of the localization using our method and AMCL. The predicate* succesfull *is* yes *when the robot localizes accurately with respect its ML hypothesis error stays localized for the remainder of the path. The distance needed to acquire localization is both expressed as actual localization of ML and as self-reported localization (by assessing convergence). Note that self-reported localization can be reported erroneously (i.e.,. false confidence). The average ML error after localization is also reported and the average number of hypotheses used both before and after self-reported localization has been achieved.*

	Scenario	A						B						C	
	Descr.	Init region						Total map						Clutter	
	Path	A1	A3	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	C11	C12
ours	succesfull	yes	yes	yes	yes	yes	yes	yes	yes	no	yes	yes	yes	yes	no
	actual dist ML	1.7	4.9	1.7	8.3	4.9	5.3	1.1	7.2	n/a	5.5	7.1	10.3	16.4	n/a
	self-rep dist	13.2	4.9	16.3	22.0	8.4	24.1	15.7	20.9	16.9	5.5	4.0	6.0	2.9	9.1
	ML avg er aft con	0.05	0.33	0.07	0.23	0.28	0.36	0.13	0.25	n/a	0.24	0.51	0.16	0.08	n/a
	max # hyp bef con.	200	200	200	200	200	200	200	200	200	200	200	200	200	200
	max # hyp aft con.	7	9	7	10	7	5	33	11	n/a	8	7	5	6	n/a
AMCL	succesfull	5/5	5/5	2/5	0/5	5/5	0/5	0/5	0/5	3/5	4/5	4/5	3/5	2/5	5/5
	actual dist ML	5.2	6.2	3.9	n/a	9.3	n/a	n/a	n/a	6.5	10.3	1.2	6.3	13.1	8.0
	self-rep dist	5	6.9	6.4	n/a	13.3	n/a	n/a	n/a	8	12.2	2.9	7.4	12.9	8.7
	ML avg er aft con	0.19	0.32	0.18	n/a	0.29	n/a	n/a	n/a	0.2	0.17	0.25	0.36	0.42	0.22
	max # hyp bef con.	5e4	5e4	5e4	n/a	5e4	n/a	n/a	n/a	5e4	5e4	5e4	5e4	5e4	5e4
	max # hyp aft con.	5e4	44217	5e4	n/a	5e4	n/a	n/a	n/a	17378	28235	13527	5e4	5e4	5e4

evaluate all of them in light of the next pairing. This, however, shows a clear limitation in terms of hypotheses needed to cover larger maps in terms of feature count. One way to deal with this is to postpone evaluation and favor the evaluation of more unique features first (i.e., temporally out-of-sequence) such as large line segments, which we currently do not do. Furthermore, maintaining hypotheses on actual object-level or even spatial object patterns can significantly reduce the amount of hypotheses and computation time (as will be shown in Chapter 4). Of course this computation time has to be balanced with the increased localization success that our method can achieve. Another drawback, in our experience, is the added complexity of our method over, e.g., Monte Carlo approaches. Our method requires bookkeeping of a local map, a feature detector and a hypothesis tree. Each step introduces possible failure points both inherent to their underlying assumptions and due to programming complexity. Furthermore, because we use features, we cannot localize when all features are obscured. For some environments that are very cluttered this would be a likely scenario and describing the environment in terms of our feature shapes may not be feasible. Some of these drawbacks are handled more effectively by scan-based approaches that match features extracted specifically for the purpose of quick association retrieval from the same scanning device. However, these methods do not offer the same abstract notion of localization and cannot deal with existing geometry, suggesting possible complementary strengths in for example the area of keeping maps up to date. Our method provides the most opportunities in cases where prior existing maps are available and multi modal perception based semantic feature information has to be fused to localize a robot both effectively and in a semantically explainable way. This allows the robot to

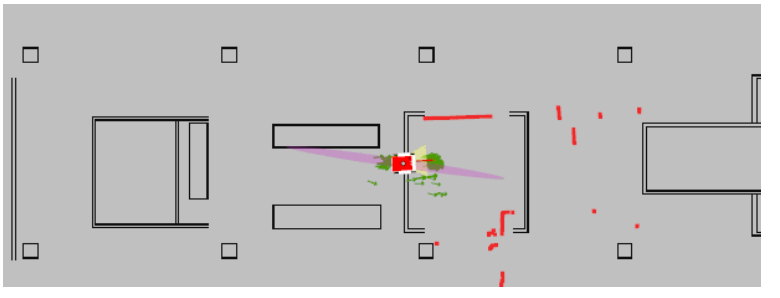


Figure 3.11. *The AMCL method failing to converge, and providing a number of wrong hypotheses while including a maximum likelihood estimate that is incorrect and intersects with occupied geometry. The red square in the center is the robot position estimate and the green arrows are particles. The LiDAR points are shown in red as well.*

update which (semantic) features are salient and which should not be used in order to keep long term localization reliable and effective.

3.8 Conclusion

We showed a localization approach that uses a local feature map and a hypothesis tree to solve the localization problem in the association space. We compared our method with a grid based particle filter and showed that our method is able to localize in more cases, while in some cases the particle filter is more successful. The grid based particle filter does not represent hypotheses on a feature association level and requires a large amount of particles in order to localize successfully in larger environments. The results are dependent on the initial sampled distribution, which causes varying results in more difficult cases. Because the environment contains unmapped clutter, convergence can not be guaranteed. In theory, increasing the initial particle count could improve results, however with the current 50,000 particles we already run into the limits of processing capabilities. On the other hand, the method that we propose, association-based localization, does not rely on such an initial sampling and performs better in some of the scenarios that we investigated, if the amount of clutter is again limited. A benefit of our method is a significant decrease in the amount of hypotheses necessary to represent the robot's belief about its location. And while our hypotheses are more expensive to maintain individually than the particles in a particle filter, the conceptual advantage of needing much less of them is promising. We foresee benefits when we consider the interpretation of the hypotheses by other parts of the motion stack (e.g. active localization planners) and the insight that our approach generates into the assumptions underlying the localization effort.

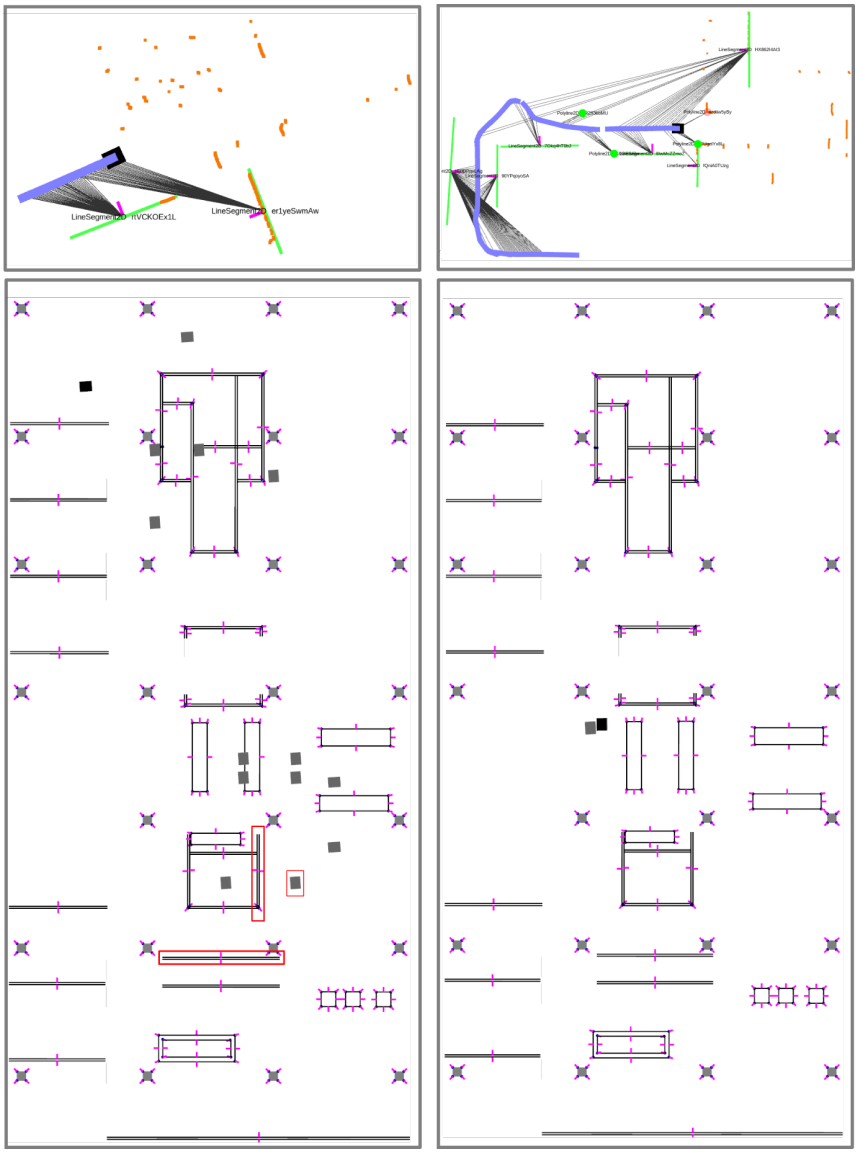


Figure 3.12. *The bottom left map shows the global hypotheses that are created based on the local map at the top left. The correct hypothesis and feature pairings are indicated by red squares. The local map shows the LiDAR points in orange and the local map in green. The right figures show the global and local map after localization is achieved. The ML hypothesis is shown in black and a number of* alternative *hypotheses in gray. One of them is significantly drifted because of a different assumption made recently. The others are covered by the ML hypothesis.*

Figure 3.13. *Three situations in which the local map contains objects that are not on the global map (marked by red circles). The camera images from the robot's camera have been logged for verification purposes. The first situation is scenario C12 where disturbances have been added on purpose. The second scenario is B5 where the scene is disturbed by pedestrians walking in front of the robot. The third scenario is B10, where the robot leaves a room and spots a dustbin and an open door as corner features. Both are not on the global map and require sufficient not-on-map assumptions to survive the pruning process.*

3.9 Future Work

Our work aims to provide robots with semantically explainable autonomy in environments for which prior maps are available. The next step is to include the robot's actions into a task-based framework for fully autonomous navigation and recovery. We see two possible directions for extending the current work: 1) Pre process the map and use higher-level salient feature combinations (e.g. on object level) to improve performance and robustness and scale up to larger environments. 2) Use the association space of hypotheses in an explicit action selection policy. The first direction considers exploiting semantic and topological relations on the local map first, and creating indices into the global map to efficiently search for feasible locations based on these relations. These relations can also incorporate free space (e.g. a single column surrounded by a large area of free space) as this provides a very strong indication of feature location when seen in the LiDAR sensor. The second point is complementary, in the sense that actions of the robot can help to first gather this feature context (active sensing) before deciding to expand the hypothesis tree. In conclusion, our work provides a step in the right direction towards recoverable and semantically insightful robot navigation that can exploit information from multiple sensors locally first.

Chapter 4

From Features to Object Patterns for Indoor Robot Localization

Abstract
One of the major challenges in robot localization is differentiating between relevant local observations of objects on the map and clutter. In this chapter, we propose a hypothesis-based robot localization algorithm that makes the distinction by looking for local *patterns* based on the assumption that the structure that appears in patterns has a much lower probability of appearing in the clutter too. The major structure we propose occurs due to the grid-spaced layout of buildings. We use a hierarchical map that models collinear and L-shaped patterns found in grid-arranged concrete columns. Our algorithm detects these patterns in a local map that is built from sensor data. The first contribution of this chapter is the just-mentioned “semantic layer" of building-specific object patterns. The second contribution is found in prioritizing the semantic patterns over individual features in the data association, where patterns are found in a locally consistent map that considers a horizon of sensor data. And finally, we introduce an elementary local “active sensing” motion to discriminate among multiple candidate patterns in the case of repetitive building structure. The generic approach is experimentally demonstrated on the ground floor of a university building, with open and cluttered spaces, with a robot driving autonomously, and a map that only contains the static building features visible to a planar LiDAR.

This chapter is based on:
R. W. M. Hendrikx, H. Bruyninckx, H. L. Chen, *et al.*, “From features to object patterns for indoor robot localization”, *(Submitted for publication)*, 2023

4.1 Introduction

Many environments contain structural objects in repeating patterns that are not (always) visible from a single pose, but are easy to discriminate from spurious clutter. They require motion actions before they are observed. Furthermore, in large maps, direct evaluation of single primitive features may be infeasible due to constraints on evaluation time or memory, either due to the amount of associations possible (especially when clutter and features have similar appearance) or due to the amount of pose hypotheses needed in the case of Monte Carlo localization (MCL) methods. Delaying evaluations while performing actions to gather more information on these patterns is an important trade-off to make when dealing with finite computing resources. Furthermore, a representation of the global map should allow the robot to switch between the evaluation and association of patterns and individual features, as the ambiguity of the environment and the amount of possible hypotheses changes. Symbolic representations of the feature patterns allow the robot to configure its localization algorithm and allow operators control over which unique parts of the environment are used by the robot to infer its location.

4.1.1 Approach

The approach is visualized in Figures 4.1 and 4.2. The first step in our approach is that we associate and combine features detected by our on board sensors into a *local map* of features, similar to Chapter 3. We proceed now with the example of features extracted from a planar LiDAR scan, i.e., lines, circles and corners. We maintain a consistent moving-horizon estimate of the feature coordinates in the local map by associating measurements from different robot poses and performing state estimation using a factor graph. We then evaluate the features on this local map for pairs and for triples that form collinear and right angles. Figure 4.2 shows a pair of circular columns used as a pattern instance. These pattern instances, together with their relative dimensions, are checked for correspondence with pattern instances on the global map. The proposed patterns have two benefits: 1) They avoid the combinatorial complexity of relating observations of primitive features to the map and 2) They make the candidate associations from the local map less sensitive to clutter (increased detection precision). In this chapter, we choose the patterns based on their detectability, which is dependent on the specific sensor and environment. Our method differs from other methods in that we do not require the scene to be visible from a single sensor frame. As soon as an initial pattern is found, the *relative dimensions* (e.g., the distance between columns) allows for quick evaluation of possible candidate patterns on the global map. If candidate patterns are found, an inspection action is performed relative to the local pattern features, that finds other features on the map in the vicinity of the pattern (Figure 4.2). This behavior is a simple `rotate` behavior in the demonstrated case, that is performed after the robot has positioned itself between the features that form the pattern. While performing the rotation, we evaluate an association tree which is initialized with

the pattern. In the evaluation of this tree, we use all matching primitive features in the environment of the pattern, as well as a free space check that we will introduce later, to effectively find which pattern on the global map is supported. The result is that multiple hypotheses are initialized. These hypotheses are then tracked and possibly falsified using matching features and free space, while the robot performs its navigation task based on the best hypothesis. The scope of this chapter includes the introduction of the feature pattern layer as a symbolic layer of feature relations on top of an existing vector map and the introduction of a generic primitive rotation behavior to inspect the pattern surroundings. Furthermore, we incorporate the patterns into a hypothesis tree and show how such a tree can be represented. Our work does not provide feature patterns for all possible use cases encountered in practice, which is a relevant subject but is deferred to future work. We also do not focus on motion planning or optimal decision making under multiple hypotheses; rather, we show that, depending on the environment, simple motion behaviors can be sufficient to find discriminating patterns and thereby significantly reduce the amount of hypotheses that are initialized.

4.1.2 Contributions to state of the art

Our work contains the following contributions to the state of the art:

1. We add a semantic layer of feature patterns that increases the robot's ability to differentiate between clutter and relevant features.
2. We prioritize feature patterns in making data associations.
3. We increase the time scale of detections from instantaneous measurements to a local map.
4. We propose an elementary active sensing motion to discriminate among patterns.

Our work is experimentally validated in an indoor environment that contains a significant amount of clutter. In this validation we use both binary patterns and composite patterns of concrete columns that exploit the grid-like structure of the environment. We also show how the pattern search procedure can be restarted as part of a recovery procedure during the tracking phase.

4.2 Related Work

Robot localization is an active field of research. Other works have relied on similar features for multi-hypothesis global localization [68], [4], [56]. We rely on the local map and association tree that we described in [57]. The authors in [4], [56] also evaluate spatial relations between features as part of a tree search. They do not, however, impose a hierarchy on the spatial relations on their global map and do not incorporate motion in their framework, although

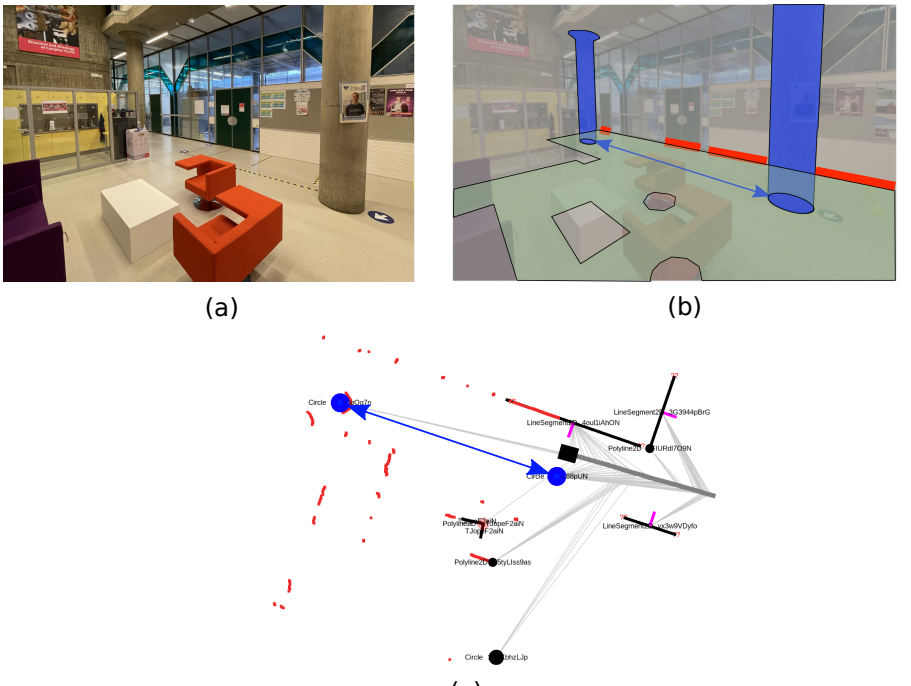


Figure 4.1. *The example we use in this chapter of how feature patterns can be used to discriminate structural features from clutter objects. A binary feature pattern in two concrete columns is shown in (a) and highlighted in blue in (b). Other relevant features in its vicinity are shown in red and free space is shown in green. The local map made by the robot is shown in the (c), where the same pattern is highlighted in blue and LiDAR points are shown in red.*

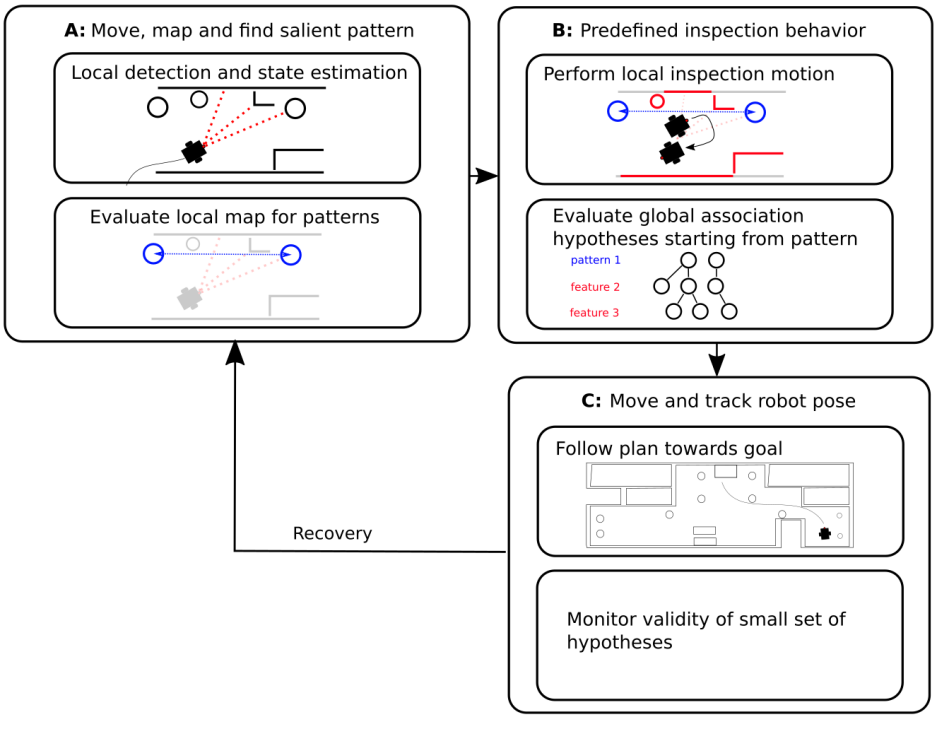


Figure 4.2. *A schematic depiction of the approach that we propose in this work. First, the robot maintains a local map of features (Figure A). The association with the global map is delayed until a pattern candidate is found in this local map. Then, in Figure B, The pattern is used at the first level of an association tree and a simple inspection motion is performed to incorporate the remaining part of the environment in the association tree search. Finally the hypotheses are tracked and monitored (Figure C). If all hypotheses are falsified we reinitialize the search for a salient pattern.*

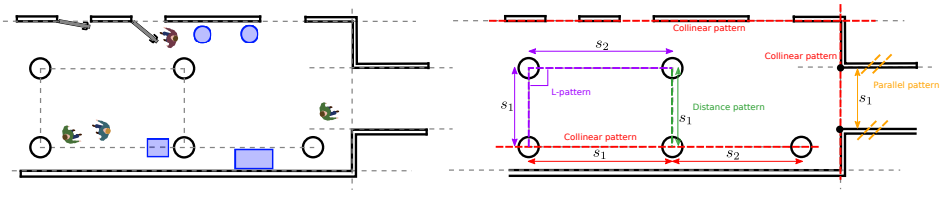


Figure 4.3. *Many indoor environments have their structural features layed out in a grid-like spacing (left image), leading to constraints such as collinearity and orthogonality (right image).*

[56] does group line features based on their visibility from the same room to increase search efficiency. The authors in [49] define a feature descriptor on multiple line-endpoint features from reference key scans in a data set. However, their descriptor takes a set of line segments with arbitrary orientation into account, while our method specifically annotates collinear and right angle relations between features that are not related to a set of key scans. Our approach considers a local horizon and thereby permits multi-scan feature descriptors in environments that contain grid-like structure. Furthermore, our approach initially disregards local features that do not form patterns to increase robustness. The authors of [68] also rely on feature based representations of doors, lines and point pairs. They introduce creative and supportive features, where only the former initiate pose hypotheses that are tracked by Kalman filters. They choose actions based on the most probable hypothesis, that maximizes the amount of new features seen. Our approach has similarities to theirs but we add a layer of discriminating patterns on top of our representation and relate motions to these patterns. Our work assumes that the robot can perform motion prior to detecting features and when inspecting a pattern to collect more information for localization. The planning of optimal motions in the face of uncertainty has been studied in literature using the Partially Observable Markov Decision Process (POMDP) framework ([119], [43], [110], [91], [76]). While optimizing the actions for localization is promising, the downside is computational complexity that prevents scaling to larger environments. Furthermore, many methods rely on grid maps or pose-based particle sets that do not offer the object-based notion of localization that we use in our work. We do not focus on highly self-similar environments, which may require additional planning or heuristic policies to be dealt with. Hierarchical map representations have also been considered in other work, most notably the addition of topological relations as a layer on top of metric information has been investigated in works such as [79], [70], [40], [77], [53], [71], [8]. Hierarchical models of indoor environments have also been investigated in other domains such as in [140], where a visibility graph is maintained for doors, corridors and intersections. These models are potential sources of information for the algorithm that we propose. The main difference with the works mentioned in this section is that we explicitly add a layer of feature patterns to our map and we only evaluate the rest of the environment after a pattern has been found. This way, our approach enables the exploitation of discriminating structure in the environment that is supplied *a priori* via a hierarchical map.

4.3 Methodology

4.3.1 Feature patterns

We denote the set of primitive features on the global map by $\mathcal{M} = \{m_1 \dots m_{n_m}\}$ which consists of n_m symbolically identifiable geometric features for a specific sensor. We will now proceed with our example of a LiDAR sensor, although other sensors may also permit suitable features. These primitive features are of type *line*

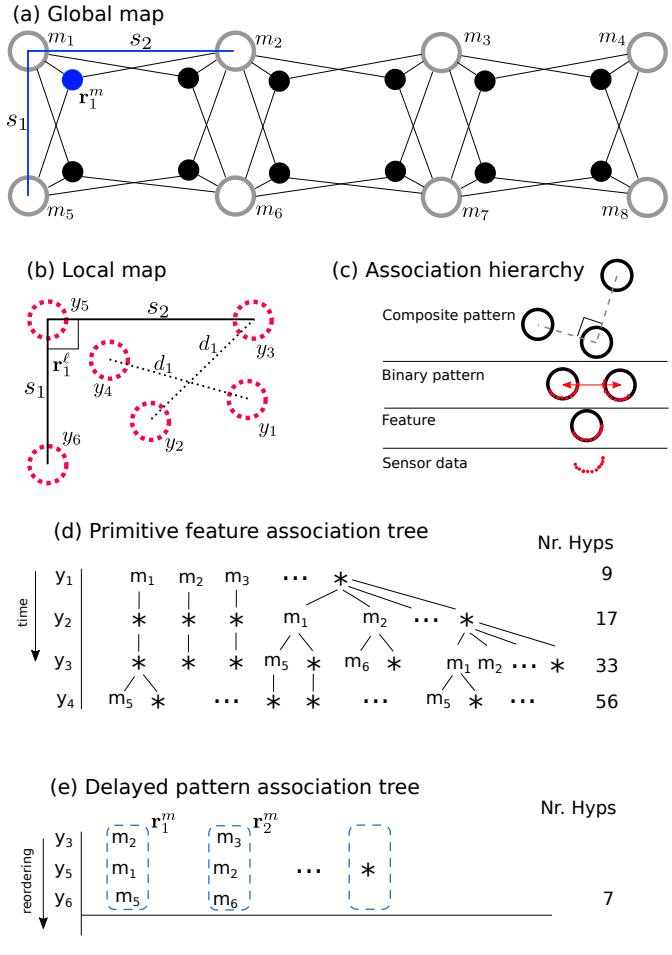


Figure 4.4. *Searching for feature patterns that originate from a grid based layout in the environment. (a) A map of the environment. One of six similar pattern occurrences is shown (b) A local map of circular features and an L-shape detected in the sensor data. As feature patterns are composed of more individual features, the chance of being resembled by clutter becomes significantly smaller. (c) The association hierarchy. (d) Hypothesis tree expansion based on individual features and on pattern features.* Not-on-map *associations are indicated by* $*$. *The horizontal levels in the top tree are formed by evaluating local features chronologically as they are observed. (e) Patterns we find in the local map reduce the amount of hypotheses, as they form composite features.*

segment, *corner* and *circle*. We add a set of feature patterns $\mathcal{R} = \{\mathbf{r}_1 \dots \mathbf{r}_{n_r}\}$ to this map, that reference the primitive features. A feature pattern $\mathbf{r}_i = (\mathbf{m}^i, \texttt{con}^i, \mathbf{s}^i)$ is a triple consisting of an ordered set of feature references $\mathbf{m}^i$, together with a pattern constraint $\texttt{con}^i$, which we restrict to `binary`, `collinear` or `L-shape` in this work. Furthermore, we add the relative dimensions $\mathbf{s}^i = \{s_1^i \dots s_n^i\}$ between the individual features to the pattern. The features we assume and use in the remainder of this work are circular shapes, although similar representations are possible for e.g., line segments and corners if appropriate interpretations of $\mathbf{s}^i$ are specified. The dimensions for a circle pattern are simply the distance between two consecutive circles in the pattern. An example is shown in Figure 4.3, where some of the patterns are shown that occur in indoor environments due to the fact that many structural features are designed on spatial grids. The local features that are mapped in the vicinity of the robot by its sensors are denoted by $\mathcal{Y} = \{y_1 \dots y_{n_y}\}$. The feature types are the same as in the global map. We will use $\mathbf{r}^m$ and $\mathbf{r}^\ell$ to refer to patterns on global features and on local features, respectively. As our local map is built, new features from sensor data become available in a queue which is processed for the occurrence of local patterns.

4.3.2 Rationale

Consider an environment that contains eight circular features which are spaced on an even grid consisting of two rows with four features each (Figure 4.4a). We would like to initiate a small set of association hypotheses that is likely to contain the correct hypotheses. A total of twelve L-shaped signatures are added to the global map a priori. Because of the uneven grid spacing, two sets of six similar patterns exist (An L-shape and a mirrored L-shape). The local map that is shown (Figure 4.4b) contains six features. Within this map we can find binary patterns and composite patterns (i.e., patterns consisting of multiple binary patterns). To evaluate which features on the global map correspond to locally observed features, we could use an association tree where each level i of the tree corresponds to a locally observed feature y_i, similar to Section 3.4.2. The nodes on that level correspond to a pairing with a global map feature and such a node is only added if it is consistent with the associations that its parent nodes represent. In this case, consistency means that the local spatial distances between features are the same up to some threshold accounting for spatial uncertainty. Because we can detect clutter or spurious feature extractions, a $(*)$ pairing is used to account for the not-on-global-map possibility. If we were to evaluate our association tree for every individual feature y_i, we would generate a large amount of hypotheses, especially if we have to account for many $(*)$ options, as can be seen in Figure 4.4d. A consequence of imposing a pattern hierarchy on the map, is that we can evaluate patterns in the local map, and initialize a first layer that is much less likely to resemble clutter. Furthermore we do not need to maintain clutter options on the dense primitive feature level, resulting in less hypotheses having to be maintained and evaluated. This is shown in Figure 4.4e. Feature patterns are more discriminating, assuming

that the robot is able to encounter them early and detect them reliably. To deal with their non-uniqueness, our approach evaluates the subsequent layers of the tree based on individual features in the pattern's surroundings. We thereby split localization in the search for a pattern that is easy to distinguish, followed by an action to determine *which* of the non-unique patterns on the map it has detected by looking at its environment.

4.3.3 local pattern extraction

Our algorithm first checks whether local binary patterns can be formed that satisfy any existing distance requirement on the global map for those primitive features, without maintaining any associations yet. These binary patterns are pushed to a stack of local candidate patterns, which are also checked against new local features. A new feature then completes a candidate triple pattern, if any of the shape requirements is satisfied (`collinear` or `L-shape`). Only in that case, the pattern dimensions are checked against an index of pattern dimensions on the global map and if at least one match is found we initialize an association tree. The procedure for extracting triple patterns is shown in Algorithm 3. The resulting triple patterns adhere to the constraints, but their dimensions $\mathbf{s}$ still have to be evaluated against the global map. During the constraint check in Algorithm 3, we check the angle θ of the triple for deviation from a right angle ($\epsilon_{\perp}$) or straight angle ($\epsilon_{|}$):

$$\theta = \cos^{-1} \frac{q_1^T q_2}{|q_1||q_2|}, \tag{4.1}$$

where we use:

$$q_1 = \begin{bmatrix} \bar{y}_1^u - \bar{y}_2^u \\ \bar{y}_1^v - \bar{y}_2^v \end{bmatrix}, \quad q_2 = \begin{bmatrix} \bar{y}_3^u - \bar{y}_2^u \\ \bar{y}_3^v - \bar{y}_2^v \end{bmatrix} \tag{4.2}$$

with $\bar{y}_2^u$ and $\bar{y}_2^v$ being the two coordinates of the circle in the middle. This means that we have to assign each of the two circles of a binary pattern to $\bar{y}_2$ and check (4.1) for both assignments, as both ends possibly form the center of a completed triple pattern. This is done in the functions `is_L_pattern()` and `is_Col_pattern()` in Algorithm 3.

4.3.4 Association tree

We employ an association tree similar to Section 3.4.2, which we modified to accommodate our feature pattern approach. The hypothesis tree consists of nodes and levels, where a node n_{ij} is on level i, signifying that it makes a data association between a local set of features and a set of global features on the map. This set of features can contain only a single feature or it can contain multiple features (i.e., supporting the pattern use case). By $\mathbf{y}_i$ we denote this ordered set of local features and by $\mathbf{m}_{ij}$ an ordered set of features on the global map. A node represents a data association $\mathbf{D}_{ij}$ that assigns a global feature to every primitive feature in $\mathbf{y}_i$ via

Algorithm 3 The procedure of evaluating local features for patterns that adhere to the proposed constraints

Input: processed features $\mathcal{Y}_{\text{p}}$, new features $\mathcal{Y}_{\text{new}}$,
binary patterns $\mathcal{R}_b^{\ell}$, triple patterns $\mathcal{R}_{\text{tr}}^{\ell}$
find_local_patterns($\mathcal{Y}_p, \mathcal{Y}_{\textbf{new}}, \mathcal{R}_b^{\ell}$):

```
for y_new in Y_new do
    for (y_a, y_b) in R_b^ℓ do
        if is_L_pattern(y_a, y_b, y_new) then
            R_tr^ℓ = R_tr^ℓ ∪ Pattern(y_a, y_b, y_new)
        end if
        if is_Col_pattern(y_a, y_b, y_new) then
            R_tr^ℓ = R_tr^ℓ ∪ Pattern(y_a, y_b, y_new)
        end if
    end for
    for y_a in Y_p do
        if exists_in_global_patterns( dist (y_a, y_new)) then
            R_b^ℓ = R_b^ℓ ∪ Pattern(y_a, y_new)
            for y_b in Y_p do
                if is_L_pattern(y_a, y_b, y_new) then
                    R_tr^ℓ = R_tr^ℓ ∪ Pattern(y_a, y_b, y_new)
                end if
                if is_Col_pattern(y_a, y_b, y_new) then
                    R_tr^ℓ = R_tr^ℓ ∪ Pattern(y_a, y_b, y_new)
                end if
            end for
        end if
    end for
end for
Y_p = Y_p ∪ y_new
return R_tr^ℓ
```

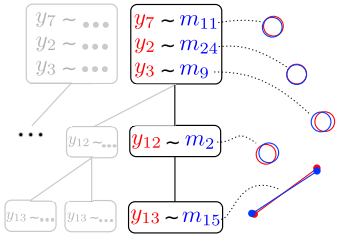


Figure 4.5. *The hypothesis tree with the first level containing a set of local features. A hypothesis considers the associations between a set of local features and a set of global features, which is evaluated for spatial consistency.*

$\mathbf{m}_{ij}$, via corresponding indices. Note that $\mathbf{D}_{ij}$ can be used to associate individual features, by letting $\mathbf{y}_i$ and $\mathbf{m}_{ij}$ contain only a single feature. Furthermore, a data association $\mathbf{D}_{ij}$ can assign $\mathbf{y}_i$ to $(*)$, implying that that specific set of features does not correspond to features on the global map. Note that this does not say anything about whether a subset of $\mathbf{y}_i$ is represented on the global map. Every leaf node represents a hypothesis by taking into account all its parent nodes which jointly assign a set of global map features to local features. The likelihood of this hypothesis is determined in a similar way as in Section 3.4.3 by using the spatial lack-of-fit that remains after performing a spatial registration for a horizon of local and global features. Determining this spatial registration is straightforward because each hypothesis fixes a set of data associations. If, after the spatial registration, any of the feature coordinates have a spatial error that is larger than a prior defined threshold, we omit (prune) that hypothesis immediately. The feature patterns that were described in the previous section are incorporated in this framework by supplying the candidate sets $\mathbf{m}_{ij}$ for $\mathbf{y}_i$, Once a spatial pattern $\mathbf{r}_i^\ell$ is found locally, its distances $\mathbf{s}_i$ are checked against a list of global patterns. If matching patterns are found with the same dimensions, a layer is added for $\mathbf{y}_i$ together with nodes for all $\mathbf{m}^k$ from the global patterns $\mathbf{r}_k^m$ that match. This pattern layer then instantiates the first layer of the tree. The consecutive individual features are then evaluated as single features (similar to our earlier work), as is shown in Figure 4.5. In this process we, assign lower likelihoods to hypotheses that contain features that cannot be matched globally and remove hypotheses for which we expect to detect features that are not seen by evaluating the local free space. In this way, we favor the hypotheses for which features are matched (true positives) and remove those that fail to see geometry where they must do so (false negatives).

4.3.5 Hypothesis evaluation and tracking

Because maintaining a local map can be error prone for some feature types, we maintain a local map until a globally supported pattern is found. Since our inspection behavior evaluates the association tree conditioned on the local-to-global pairings of this initial pattern, we move to a location approximately between the

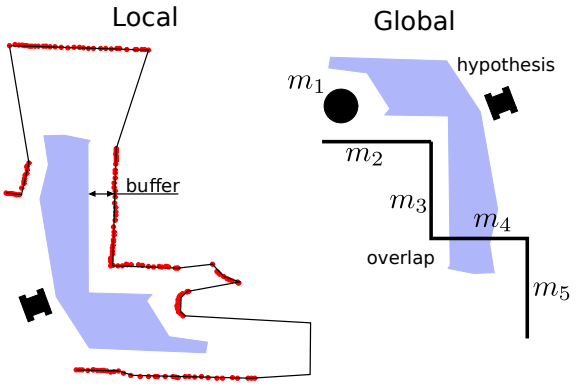


Figure 4.6. *Free space polygon resulting from simplifying and buffering a polygon formed by the LiDAR points. The result is checked for overlap against global features.*

pattern features. This pattern is then partially visible during a rotation, serving as a reliable reference for the spatial congruence check. Before rotation and tree expansion, the local map is cleared, except for the features that form the local pattern.

4.3.6 Falsification using free space

Absence of expected geometry is a strong indication that a hypothesis is incorrect. Hence, we use free space from a LiDAR scan to check for overlap with geometry on the global map. First we form a polygon by applying split-and-merge to the LiDAR points that fall within a defined scanning radius and close it using the LiDAR device origin. Next, we shrink the polygon with a negative buffer distance such that the polygon moves away from its edges (using the Boost geometry library [15]). This approach also shrinks laterally from potential hit-points that would otherwise be very sensitive to small rotational errors of the robot pose. We generate this polygon once locally, and then check for overlap using the pose for each hypothesis during the association tree search (see Figure 4.6). The shrink distance is chosen sufficiently large such that openings in walls that are not on the map (such as doors or glass strips) do not appear as free space.

4.3.7 Tracking of hypotheses

After the inspection of the local pattern, the association tree evaluation has generated hypotheses for the individual features in the map. Because we split hypotheses using $(*)$ associations, we get clusters of hypotheses at every occurrence of the pattern in the local map. For each pattern occurrence, we maintain the hypothesis with the largest likelihood and switch to a tracking approach (i.e., we never keep more than one hypotheses for a pattern occurrence). We let the robot move

autonomously towards the goal location, assuming that the hypothesis with the highest likelihood is correct. Each hypothesis is tracked by associating global features with the local features found in the LiDAR scan and using a factor graph to update the pose estimate in the map. We apply the same free space as described in the previous section to each hypothesis individually to remove hypothesis that violate expected local free space. Furthermore, we update the likelihood of the hypotheses based on the lack-of-fit of features in each scan, similar to Section 3.4.3. Our mechanism for a change in executed behavior is based on two events: 1) A different hypothesis that is tracked gets highest likelihood, in which case our robots motion will change accordingly. 2) No more plausible hypotheses remain based on the free space check, in which case we trigger our recovery behavior. Our recovery behavior is indicated in Figure 4.2, and amounts to starting the search behavior again. The robot will first rotate and the drive around until a pattern has been found again, based on which new hypotheses are generated that can be tracked again.

4.3.8 Motion primitives

Our motion behaviors consist of a kinematic trajectory predictor for pairs of forward and rotational velocity (v_x, ω_z). The robot footprint is forward predicted for a time horizon (three seconds) according to a set of velocity pairs and the resulting future footprints are checked for overlap with LiDAR points, in which case the specific (v_x, ω_z) does not satisfy the collision constraint. For the search behavior, we choose feasible trajectories that go forward, while for the inspect and goal-oriented behaviors we supply a way point to a local point between two pattern features, and to the goal location, respectively. We make the assumption in this work that the space between two features that form a pattern can be reached by the robot, to keep the motion planning simple, while noting that this may be too simplistic for other environments.

4.4 Experimental Validation

In this section, we evaluate our proposed method in part of the Gemini building on the Eindhoven University of Technology campus which measures 70x17 meters. It consists of a large open space connected to a hallway that is intermittently occupied by building elements. The environment contains two rows of circular columns that, together with the walls of the building, form the features that are available on the global map. Figure 4.9 shows the environments. All move-able objects, including large tables and sofas, are moved regularly to accommodate the changing needs of building users. Therefore, our robot platform, which is shown in Figure 4.9, has to rely on the structural features only. Additionally, our circle detection approach will sometimes trigger on obstacles and bystanders, which is a scenario that our approach is meant to mitigate. To evaluate our approach, we consider the scenario where the robot has to drive to a goal location without having knowledge of its

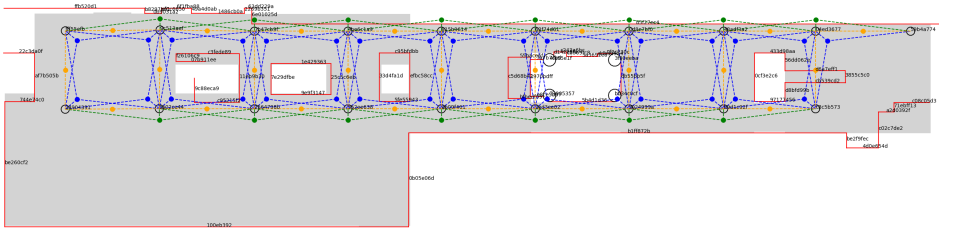


Figure 4.7. *The feature map of the environment in which we test our approach. The circular columns in the map (black) are members in three types of relations: binary relations (yellow), collinear triples (green) and L-shape triples (blue). All other geometry in the environment is shown in red and annotated by unique ids. The relation identifiers have been omitted. The grey area denotes the workspace that contains valid hypotheses.*

initial position. We consider autonomous driving important during our experiment, because the patterns should be detectable without having any notion of location available. We consider 6 different positions from which the robot will start, as indicated in Figure 4.9. We consider two different sets of feature patterns that are added to the global map and evaluate the result of our strategy on both sets. The first set consists of only binary feature relations between neighboring columns that contain free space between them. The second set consists of collinear and L-shaped feature relations between triples of columns. The feature sets are also shown in Figure 4.7.

4

4.4.1 Generating feature relations

The feature relations are generated by a Python script that takes as input the map of the environment and evaluates for neighbouring columns for which the connecting line does not intersect any geometry. An example of a column expressed in our map format is the following excerpt:

```
{
"@id": "ColumnCircular.008",
"@type": "Column",
"represented_by": [{
   "@id": "8044c69b",
   "@type": "ObjectFeatureRepresentation",
   "perceivable_by": {
      "@id": "PlanarLidar2D"
   },
   "represented_by": {
      "@id": "ab09f461",
      "@type": "Circle",
      "radius": 0.33,
      "has_placement": {...}
}}]}
```

We use the JSON-LD specification as a host language [73]. We evaluate all column

geometry for neighbouring pairs on the grid, leading to the binary patterns that contain there relative distance in the "s" field as shown below.

```
{
"@id": "bcb460",
"@type": "FeaturePattern",
"constraint": "BINARY",
"features": ["9c513aef","fff55efb"],
"s": 7.493890799037894
}
```

Note that the x- and y spacing of the column grid is not the same (7.4 and 6.2 meters. respectively). This results in different signatures, with a total of 17 similar ones for the x-aligned pairs and 9 for the y-aligned pairs. The pairs are then evaluated for column instances that are common, leading to the collinear and L-shape pattern instances. Because of the grid spacing, collinear pairs have a symmetry allowing two different pairings. For the L-shape patterns, we adopt an *L-axis* convention, where the order of columns is such that the lower-right part of the L-shape is the first column, followed by the corner point of the L and the upper end of the L. No symmetries exist for a L-shape pattern. The distances that represent the leg distances are added to s^i. The pattern representation contains symbolic references both to the binary patterns and the individual circle features as shown below.

```
{
"@id": "c4502a",
"@type": "FeaturePattern",
"constraint": "COLLINEAR",
"subpatterns": ["bcb460","342149"]
"s1": 7.493890799037894,
"s2": 7.3993377685546875,
"features": [
   "3ed3ee02",
   "ab09f461",
   "552ee638"
]}
```

4.4.2 Behavior implementation

Because we perform closed loop experiments, the robot needs behavior coordination to perform motion actions. Our behaviors are implemented inside an open source behaviortree framework [51]. The tree is shown in Figure 4.8. The robot task starts with a rotational motion to scan the environment for features. It then performs a driving motion that avoids obstacles and goes straight when possible. While these motions are performed, our localization (which runs in a separate executable) makes a local map that is evaluated for patterns. If a pattern is found, the robot drives to a location between the two pattern features and performs a rotation motion. To this end, the behavior coordination executable requests an updated local map from the localization executable. Before this motion is performed, the

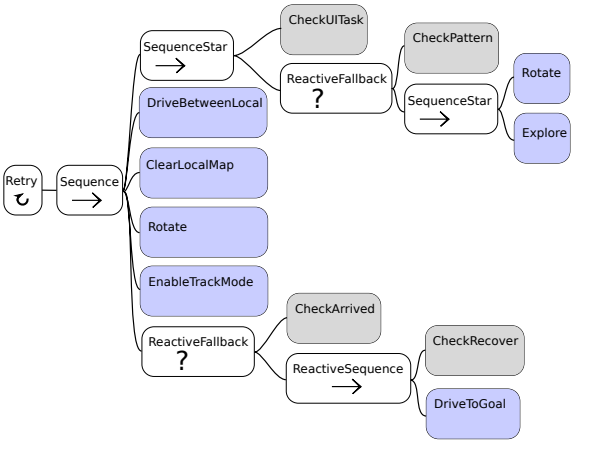


Figure 4.8. *Our implementation utilizes a behavior tree with asynchronous actions to monitor and coordinate the localization. We use the implementation from [51]. We run our approach in two separate executables, where the motion behaviors are implemented as asynchronous actions in the behavior tree, and the local mapping is a separate executable with query interface.*

coordinator clears the local map via a request from all features except the features belonging to the pattern. This ensures that the map remains *local* and no difficult loop closures must be evaluated. The pattern is added to the root of the hypothesis tree and all new features that appear in the local map are evaluated by expanding the tree, while removing hypotheses that violate the free space overlap criterium. After rotating, all remaining hypotheses are traced back to their *originating pattern* and for each *originating pattern* the hypothesis with the highest likelihood is used to track and monitor while driving to the goal location indicated in Figure 4.9.

4.5 Results and Discussion

The trajectories that the robot has driven are shown in Figure 4.10 and the results are summarized in Table 4.1. The trajectories indicate the search and inspect behaviors by dotted lines, while tracking behaviors are marked by solid lines. The appearances of (complete) feature patterns are marked by dots that are connected to the initiating features. As can be seen in Table 4.1, all but a single experiment was successful in arriving at the goal location. During the motion of the robot, we sometimes ran into problems related to the robot not seeing legs of chairs or strips of glass. To this end, we added tape to some areas of glass to prevent issues, and had to put obstacles before chairs with very slim legs. This also resulted in the correct appearance of line segments in the LiDAR readings for some of the ridges in the environments, which otherwise would sometimes be overlooked depending on the pose from which the robot observed them. Handling such sensor specific

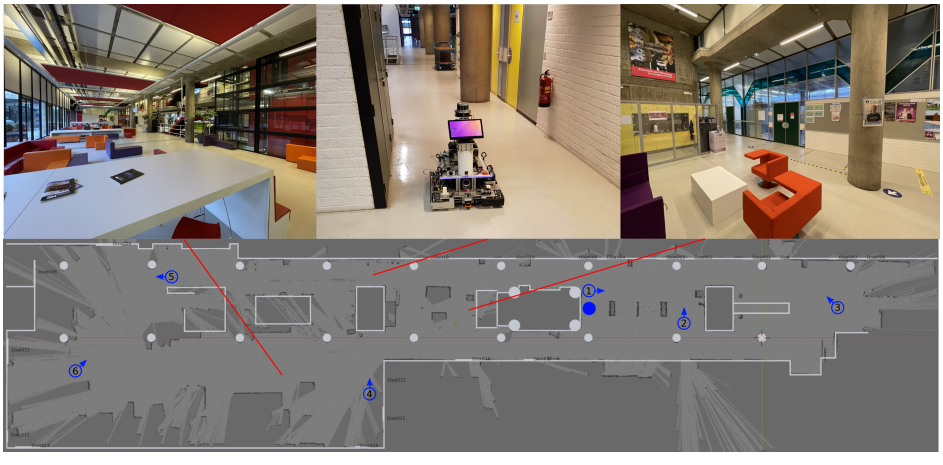


Figure 4.9. *The environment in which we will test our approach. The map in Figure 4.7 contains only the static geometry, leaving a large amount of furniture not represented on the map. The main motivation for our approach is the objective to handle such situations robustly and symbolically explicit. Overlaying is a SLAM-built grid map that shows the geometry visible to the LiDAR. The structural elements of the building that the robot has access to on its map are shown in white. The starting poses of our experiment are shown by the arrow indicators. The solid blue circle denotes the goal destination for all initial positions.*

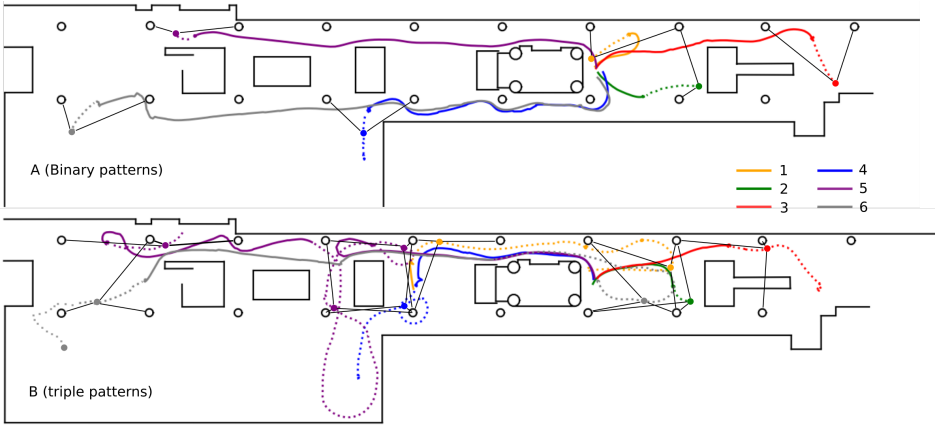


Figure 4.10. *The trajectories of the 12 experimental runs, that have been generated from the logged data afterwards. The robot poses from which a pattern is found are indicated by the dots that are connected to the pattern features. Dotted trajectories indicate explore or inspect mode, while solid lines indicate tracking mode.*

challenges is beyond the scope of our work. Videos of the localization are available online[1].

4.5.1 Pattern recall

For the binary patterns, initiating column pairs are found very quickly, mostly within the rotating phase of the search behavior. The pattern causes preemption of the rotate behavior and the robot immediately drives to the inspection pose defined locally between the columns. On the map in Figure 4.7 it can be noticed that on the right, a pair of vertical columns has no free space in between. This caused an issue for a single run (A3) which had to be restarted once, because the robot was stuck in trying to move between the columns. However, in general, the binary pattern is successful in recalling columns from a local map of multiple erroneous circle detections. This can be seen in Figure 4.11 for, e.g., run A6, where only one of the three columns is indeed truly a column. What is interesting to note is that while the robot arrives successfully in all cases for scenario A, alternative hypotheses remain three of six runs. It is due to the lower likelihood of these hypotheses that they are correctly deemed false. When we look at the triple patterns in Table 4.1, we see that they take longer to recall, as is expected. However, they still appear in reasonable time for most cases. In run B6, a collinear pattern is first detected that is indeed a column triple, but one column appears outside of the area that we mapped for this experiment, through an open door. The results is that during inspection, no feasible hypotheses remain and recovery behavior is triggered, after which a correct L-shaped pattern is found. Interestingly, during the B1, the robot fails to arrive at the goal location because another location appears too similar, as can also be seen in Figure 4.10 for that trajectory. What is not visible in that figure, is that the reason for this similarity is not the wall (which is more offset from the columns) but a couch that has been placed against the wall, making it appear similar to the goal location. During run B5 and B6, recovery behaviors are triggered twice, due to the robot looking over a ridge or through an open door from too close by. The recovery behavior does allow the robot to find a new pattern, reinitialize and ultimately drive towards the goal location.

4.5.2 Local mapping

In Figure 4.11 we also see the scenario of B1, where clutter circles can be distinguished from columns by their pattern. The grey circles indicate circles that are not spotted over a temporal window, and are thereby not considered stable detections yet. While the patterns we use are detected well in most cases, it is also important to understand the limits of the local mapping approach in the case of false detections. Our local map uses euclidean distance thresholds for associating

[1] https://youtu.be/kgZcB6oX0AY
https://youtu.be/_ufP3Z4l2Lk
https://youtu.be/jEpf9RVfA_U

local features and incorrect associations can be problematic. For example, during run B5 (last video), a large local map was maintained during the recovery behavior in which a false data association caused some distortion. These effects limit the amount of false detections that our approach can handle, however, the impact in our experiments was limited and we believe they can be further mitigated by resorting to more robust mapping strategies such as *robust error models* (see, e.g., [101]) or by selecting only more stable features (e.g. long line segments) as error contributions in the factor graph.

4.5.3 Differentiability and uniqueness

A relevant question is why the columns form strong feature patterns and whether regular spacing is important. Ideally, a pattern is not only easy to discriminate from its direct environment, but also ubiquitous enough for fast encountering and unique among other patterns. The main benefit of the triple patterns over the binary patterns, is that the spatial constraints itself are already discriminating. An irregular distance between columns (in the y-direction) would result in more unique descriptors in $\mathbf{s}^i$, making the robot less dependent on the environment for differentiating among patterns. This ambiguity in the environment is also reflected in the hypotheses that are created, which are visualized in Figure 4.12. In (a), the hypotheses that are spawned during the inspect rotation behavior are shown. In (b), the hypotheses that remain after inspection are shown, as the tracking behavior is initiated. The grey lines indicate features that are associated with the global map for the hypotheses. In (c), the remaining hypotheses are shown, where the correct hypothesis has the highest likelihood. If we used a method that would rely on all geometry in the sensor data, setting the correct lifetime parameters of hypotheses would be a difficult task. Instead, our approach makes the creation and evaluation of hypotheses very explicit. The benefits are especially visible in the areas of the building that contain much clutter, such as scenario 5 and 6 and during some of the recovery behaviors.

4.6 Conclusion and Future Work

In this chapter we showed how patterns in indoor environments can be utilized for global localization and recovery, by letting the robot focus on the grid-like structure of an indoor environment. By delaying the evaluation of hypotheses until a discriminating pattern of columns has been found, we can robustly initiate an association tree search. We also showed how we can monitor localization and trigger recovery behavior to again search for patterns if needed. The result is not only a robust approach to localization, but also one that is intuitive and explainable. During experiments, having access to the local map and the pattern that initiated the robot's assumptions on a robot-mounted display, made the process insightful and easy to explain to bystanders. The main topic for future work is generalization of this approach to other environments and sensors, and investigating the trade

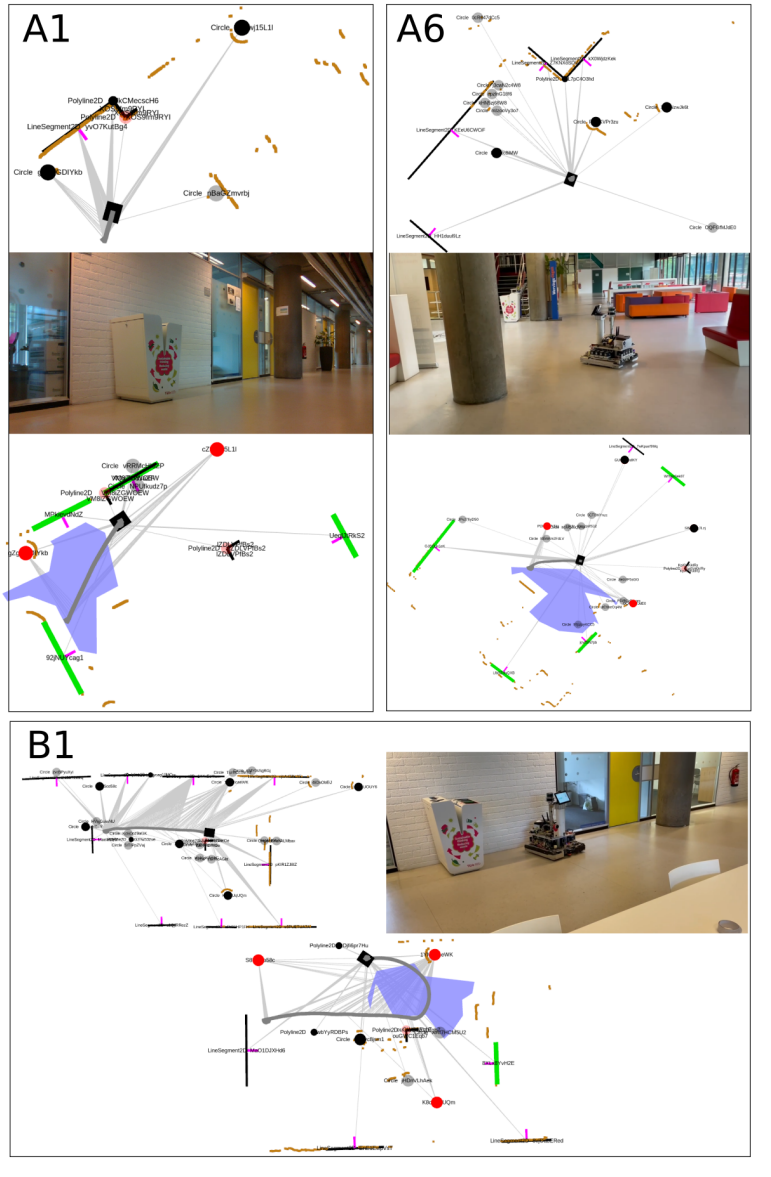


Figure 4.11. *The local maps that initiated the inspection of feature patterns for three different experimental runs. Images of the scenes are also shown. The bottom map in each frame shows the local map while inspecting the feature pattern environment for other primitive features. Lines that are highlighted in green have a match with a line on the global map for one or more hypotheses. The free space that is used to check against overlapping global features is shown by the blue polygon. Due to clutter, the size of this free space within the local LiDAR reading remains often small and the buffer distance from objects that are hit by LiDAR decreases it further. The LiDAR points are shown in brown. Circles that have been observed less than three times are shown in light grey and are disregarded. Circles that are observed more than three times are shown in black.*

Figure 4.12. *The global hypotheses for run A1 indicated by rectangles. (a) shows the hypotheses (13 clusters total) that are instantiated during the tree evaluation. The hypothesis with the largest likelihood is shown in black. (b) shows the remaining hypotheses straight after inspection, when switched to tracking mode. The grey lines indicate features spotted for the hypotheses. (c) shows the remaining four hypotheses after the robot arrives at its goal location, where the correct one has the highest likelihood.*

Table 4.1. *Results for search and inspection of binary patterns and triple patterns. The Success predicate indicates whether the robot arrives at the the target location. The pattern types found are indicated by B(inary) L(-shape) and C(ollinear). The number of circles on the local map at the time of the pattern appearing is also indicated, where the parentheses denote the number of circles including* candidate circles *that are not detected consistently over a temporal window. The (*) mark was added to the final run because a correct collinear shape was found in columns outside the mapped environment through an open door.*

	Binary pattern						Collinear / L-shape pattern					
Route	1	2	3	4	5	6	1	2	3	4	5	6
Succes	yes	yes	yes	yes	yes	yes	no	yes	yes	yes	yes	yes
Pattern type	B	B	B	B	B	B	L	L	L	L	C/L	L
Total time	0:37	0:41	0:56	1:22	1:12	2:19	na	0:39	1:22	2:30	5:19	5:29
# wrong pattern initialization	0	0	0	0	0	0	0	0	0	0	1	1*
Time to first correct pattern	0:02	0:05	0:02	0:25	0:08	0:20	0:40	0:06	0:41	0:44	0:26	1:31*
nr. circles in local map	2	3(5)	2(4)	3(8)	2(3)	4(9)	6(17)	3(4)	4(13)	3(14)	3(6)	4(12)
# hyps after inspect (correct pattern)	7	4	4	3	4	1	0	1	5	4	3	1
# hyps after arrival	4	4	2	1	1	1	3	1	1	1	1	1
# recovery behaviors	0	0	0	0	0	0	1	0	0	0	2	2

off between recall and discriminative power of different kinds of feature patterns. Furthermore, we used rudimentary motion behaviors in this work in relation to feature patterns. Localization can benefit from a stronger coupling between the local pattern and the motion, for example by defining suggested directions relative to the pattern and relative to features that form partial patterns. This way, more sophisticated behaviors can be performed that explore more rapidly for patterns and can incorporate goal-oriented planning in the exploration phase. Environments in which we expect benefits are, for example, industrial warehouses where the contents of racking change rapidly, but the racking structure can be exploited. Or corridor scenarios, where the patterns are formed by the structure of intersecting hallways. Both examples also raise the question whether free space can be incorporated into these pattern descriptions in a way that is locally descriptive and not dependent on the pose of the robot. This leads to more robust systems that can configure their behavior and sensor processing based on the structure of the environment.

Chapter 5

Conclusions and Recommendations

In this thesis, three main chapters have been presented that provide answers to the research questions posed in section 1.1.3. The contribution of these three chapters are now considered in a final conclusion, which reflects on the objectives that were formulated in section 1.3. They are followed by recommendations for future work.

5.1 Conclusions

5.1.1 Building information models and tracking

In the second chapter of this thesis, it is found feasible to use BIM models as a source of information for localization in previously unseen indoor domains. Objective 1.1 is realized by using a graph-based data model that represents the structural building items by sensor-specific features. The input for this model is generated via a conversion path that relies on IFC-JSON to find walls and columns, which are then added to a spatial database. This path relies on the JSON-LD API to query relevant IFC entities and can be adjusted for different kinds of features. In principle, this method also has access to other information in the graph, such as material properties and connected spaces. This information was not used in the demonstrator and querying this information from a graph-based IFC model is not always straightforward. More elaborate graph query languages are useful here and are explored in follow-up work [30], [99]. We were most interested in the connection to objective 1.2, which is achieved by extracting features from sensor data and using a factor graph to estimate the pose of the robot with respect to the building floor reference coordinate system. If sufficient static parts of the environment are visible to the LiDAR, the robot can successfully track its location and is able to handle small deviations that may occur due to modeling errors or missing visibility due to, e.g., glass elements or open doors. The feature length

thresholds were deliberately chosen to be robust against the latter. Modeling errors in the building, such as walls that were shifted up to 60 centimeter, are expected to be more problematic. Our method was able to recover from these errors, but is not designed to do so reliably. The level of reliability that can be achieved is evidently dependent on the accuracy of the specific building model and may increase with different sensor modalities. Robustness mechanisms are considered as part of a global localization approach in the next chapter.

5.1.2 Association-based localization with local maps

In Chapter 3, a multi-hypothesis approach is demonstrated that uses explicit associations between a local map and the global map. This method builds on the representation that was introduced in Chapter 2. Objective 2.1 is fulfilled by introducing an association tree that models hypotheses between features on a locally consistent map and the global map. The factor graph mechanism from Chapter 2 is now used to estimate a locally consistent map, which proved reliable for the type of features used (corners and line segments). As a result, the evaluation of the hypotheses (Objective 2.2) is only performed for new stable local features. The criterium used for this stability is that three detections from different poses of the same feature must exist. This mechanism was originally designed for the circles used in the next chapter and is less relevant for lines, as they can be more reliably detected from single scans. The assumption of local spatial accuracy proves justified and the evaluation of hypotheses reduces to determining the lack-of-fit after a spatial registration step, given the associations. Whether this is always the case will be reflected upon in the next section. The lack-of-fit evaluation using point registration is sufficiently fast for the map considered, as shown by the CPU time logs. The main benefit over faster constraint checking procedures (such as parameterizing the relative distance and angle between each pair of lines) is that this method minimizes the euclidean error over a set of points, which has a more straightforward probabilistic interpretation. This allows the spatial rigidity assumption to be slightly violated and also introduces some robustness to global map deviations. The performance of the method was compared to the grid-based AMCL particle filter implementation (objective 2.3), which requires a very large amount of particles (50,000) to achieve reasonable performance. The association based approach performed better in most runs and requires substantially less hypotheses (200), due to the absence of random initial sampling. A small set of strong local features is often sufficient to generate a manageable set of hypotheses. The main Achilles-heel of our hypothesis evaluation method is a scenario where too many clutter objects appear directly after each other. In this case, the N^* pruning criterion will remove the correct hypothesis in order to limit the tree growth. The main lesson learned here is that other means of pruning hypotheses must be considered to avoid the need of N^* pruning. One example might be a threshold in likelihood, above which no tracks are split, which is also suggested in multi-hypothesis tracking literature. However, determining when this non-splitting is justified based on a

spatial criterion is not straightforward, hence the current policy of postponing the decision after future observations. A more promising strategy is the use of absence of expected features (based on the global map) to mitigate tree growth. This is where a LiDAR device shows it strength and we explore this in Chapter 4, together with a feature pattern approach that selects only distinctive patterns to initiate the association tree.

5.1.3 Feature patterns

When larger buildings are considered, or when sensor observations become less reliable due to false detections or a cluttered environment, mechanisms are needed that increase the ability to discriminate. The challenges associated with this are: 1) maintaining an accurate local map and 2) managing growth of the hypothesis tree while also accounting for clutter. The latter is related to objective 3.1, which is addressed in Chapter 4 by adding a layer of collinear and L-shaped patterns to a map of the ground floor of a university building. These patterns turned the columns from relatively weak features (because of an inaccurate detector) to strong features that can be easily detected in a local map. The method focuses on the spatial constraints first, and adds a patterns - as a whole - to the first layer of the association tree (Objective 3.2). Objective 3.3 is then addressed by performing a rotation action that is defined with respect to two columns to maximize visibility of other features in the surroundings. During this step, a free space polygon is used that is very successful in pruning hypotheses, by which it also addresses objective 2.2. Sometimes the free space was too quick to prune, such as in run B5, because a large double sliding door opened which was modeled as static geometry. However, this can be avoided by removing doorposts from the layer of fixed geometry. Overall, the experimental runs show that the patterns can be reliably detected in a local map in most cases, even when substantial clutter and spurious detections are present. Binary patterns were detected substantially faster than triple patterns, and proved reliable. Triple patterns, however, are even less likely to appear in false detections. The main lesson learned from this chapter, is that while certain features may not be unique or even reliably detectable, pattern relations can turn them into strong *hyperfeatures*. The ability to dissociate these hyperfeatures is important in large environments and can also be applied to other repetitive structures such as fences, rackings or doors.

5.2 Recommendations

5.2.1 Building information models

The features and sensor used in Chapter 2 are straight forward to generalize to other indoor environments. However, there will exist BIM models that are less suitable for localization for two reasons. First, although line segments, corners and circles are relatively common features, they may not be adequate to describe all

indoor scenes. For example, some buildings consist of curved geometry. Second, the approach in Chapter 2 provides some robustness against clutter, but has to perceive enough structural features that are not resembled by unmapped objects. A next step would be to exploit more geometric and semantic class information from BIM models, such as the location of doors and stairs. Furthermore, the extension of the method to 3D LiDAR detections is one that can be performed relatively easy in the detection front-end (e.g., using *vertically stacked*, planar features), providing more accurate detections of structural items such as columns and walls. Three-dimensional information is also available in many BIM models, and can increase robustness even further. Together with a semantic detector, this will significantly increase performance in scenarios that contain more clutter. This also includes the extraction of material information (e.g., glass or reflective surfaces) and the possibility to include topological information regarding spaces and their connecting interfaces. The IFC standard provides the user freedom in choosing the representation of this information and this implies that the conversion path has to be extended and generalized in follow-up work.

5.2.2 Association-based localization with local maps

The approach we suggest in Chapter 3 and extend in Chapter 4 enables global localization and adds a level of robustness to localization that protects against additive disturbances in the model. There are, however, two other sources of disturbances that require further investigation: 1) modeling errors in the global map, as mentioned in Chapter 2 and 2) dynamic disturbances and incorrect associations in the local map. The latter was mitigated in Chapter 3 by including only features that are rarely detected in moving obstacles. In chapter 4, the circles we included would also appear in the detection of feet. In some cases this caused the local map to become distorted because of wrong associations or violation of the static object assumption. Solutions to this problem may include using robust error models that can correct data associations in the local factor graph, or other detection modalities that provide stronger features, possibly combined with semantic class information. Despite these challenges, maintaining a local map over a spatial horizon still has clear advantages because it provides context to features via patterns and provides a central model in which sensor information can be fused and monitored. However, the optimal association horizon may be smaller when only a small number of hypotheses are tracked at higher speeds or in more dynamic environments to avoid local inconsistencies. This requires further investigation into robust tracking policies for various feature types and environments.

The pruning of the hypothesis tree is arguably the most important part of the localization algorithm. The approach of allowing N_* consecutive unpairable features is attractive because of its simplicity but can be problematic if too much clutter is present (i.e., a row of chairs or dustbins). The addition of feature absence by using free space in Chapter 4 provides a better basis for pruning hypotheses and is recommended. The main challenge in the evaluation of free space is to be robust

against small errors in the pose estimate of the robot. While our maps had high enough feature density to reliably overlay this free space, cases may occur where it is better to postpone this effort due to uncertainty. Future work should investigate methods that take this uncertainty into account or evaluate free space in relation to local features.

5.2.3 Feature patterns and the semantic map

To make the method in Chapter 4 generally applicable, a set of feature pattern templates must be conceived of that exploit more of the grid structure found in walls, columns, doors, rackings etc. The inclusion of free space in these patterns is also a promising approach to navigation in symmetric environments such as hallways. Algorithms are needed for automatically generating this layer of feature patterns starting from a semantic map of primitive features. While these patterns can be composed by considering only the global map, they can also benefit from recorded sensor data. This data can be used in semi-supervised learning approaches, that aim to optimize recall and uniqueness of patterns among typical cluttered scenarios, by selecting among a large candidate set of features combinations.

A related challenge that advocates for the use of both local maps and feature patterns are inaccurate building models. BIM models can serve as input for the global localization approach in Chapter 3. However, in this thesis we were not concerned with the effect of model errors in this specific instance of a building model and corrected the errors before using the geometry. In future work, invariant structures of the building (parallel, collinear and grid patterns) can be matched to the local map in a way that relies less on the exact coordinates. The local map and local occurrence of patterns arguably provide a better basis for localization and determining actions then a pose with respect to an inaccurate map. Furthermore, such a map may lead to problems when executing a path, e.g., through a door, that is not at the exact location. A local map provides a better basis for performing this action and the connection between the local map and motion actions is a recommended topic for future work. This also includes the planning of actions that are informative, while adhering to constraints on the global semantic map regarding safety and predictability. Such a planner will benefit from the reduced amount of hypotheses that our method generates and might even (partially) precompute informative actions for locations that are known to be ambiguous.

In conclusion, the work in this thesis relies heavily on the semantic map as the central container for both the geometry and for information on how to efficiently link sensor data to this geometry. It is expected that even more semantic configuration can be realized via the semantic map, thereby further increasing the explainability and performance of robot systems. Recommended future work amounts to 1) Generalizing the representations and patterns that are relevant to model on a global semantic map, and 2) designing generic behaviors that allow the robot itself to determine which representations, both on the local and global map, are relevant

for the localization task and which are not.

Appendix A

Local Mapping with Line Segments

A.1 Local Map Building

A.1.1 measurements and factors

In this appendix we provide a more detailed explanation of how line measurements are used locally in our approach. First, to extract lines we use the implementation found in [102], [47]. A line sensor feature is represented by the point-pair coordinates $(p_{\text{start}}, p_{\text{end}})$ with respect to the robot and is extracted from a single LiDAR scan (Fig. A.1). This line gets added to the local map (in frame ℓ) as a point-pair. A second partial line segment sensor feature can be associated with the same local feature. This association is made when both points of the new sensor feature fall within a distance d_{n1} from the infinite line extending from the line on the local map and one of the points of the new line is within the segment. To solve for the most likely position of the line in the local map given multiple measurements, the point-pair representation is temporarily converted to an infinite line with range-bearing parametrization (Fig. A.2.). In order to optimize over both the robot pose and the line parameters, we create a factor for the measurement error, which is the difference between expected and measured (ρ, ϕ), given by:

$$e_\rho = \begin{cases} \rho - s + x\cos\psi + y\sin\psi, \\ \rho + s - x\cos\psi - y\sin\psi \end{cases} \tag{A.1}$$

$$e_\phi = \begin{cases} \phi - \psi + \theta, \\ \phi - \psi + \theta + \pi \end{cases} \tag{A.2}$$

where the latter case needs to be used when the robot is in the halfspace of the line that does not contain the local frame origin. Note that we need to evaluate the angle error as being a relative orientation within $[-\pi, \pi)$. To optimize over the

A

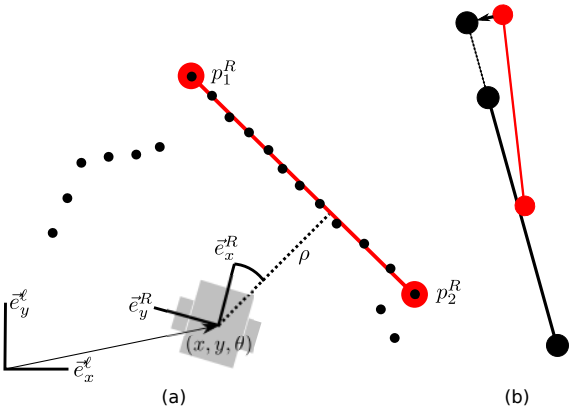


Figure A.1. *(a) The robot measures a sensor feature locally in the form of a line segment. The line segment gets added to the local map and a range-bearing measurement is added as well. (b) When a new sensor feature is associated with the same line segment on the local map, it is merged to form updated end points. The new range bearing measurement is also added between the robot pose and the line (but not shown).*

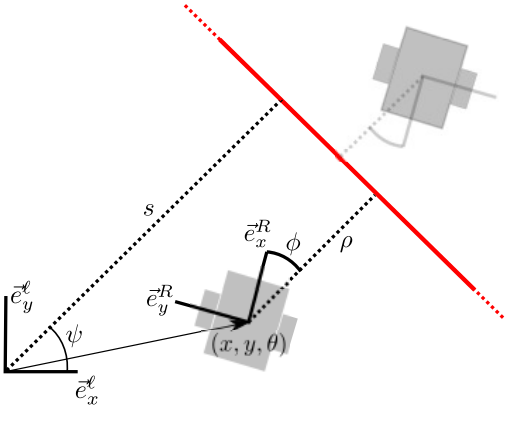


Figure A.2. *When the line segment is updated in the state estimate optimization, first a temporary range-bearing parametrization is generated in the local map as well to simplify calculation of measurement error and error Jacobians. Two possible cases are shown for the robot pose with respect to the line.*

A

robot pose and line parameters, we obtain the following jacobians:

$$\frac{\partial \mathbf{e}}{\partial \mathbf{x}} = \begin{cases} \begin{bmatrix} 0 & 0 & 1 \\ \cos\psi & \sin\psi & 0 \end{bmatrix}, \\ \begin{bmatrix} 0 & 0 & 1 \\ -\cos\psi & -\sin\psi & 0 \end{bmatrix} \end{cases} \tag{A.3}$$

$$\frac{\partial \mathbf{e}}{\partial \mathbf{y}} = \begin{cases} \begin{bmatrix} -1 & 0 \\ -x\sin\psi + y\cos\psi & -1 \end{bmatrix}, \\ \begin{bmatrix} -1 & 0 \\ x\sin\psi - y\cos\psi & 1 \end{bmatrix} \end{cases} \tag{A.4}$$

again depending in which halfspace the robot is located. The implementation in GTSAM [32] requires a custom implementation of the 2D line type with manifold traits and a custom factor. The type needs to have the interfaces `retract()` and `localCoordinates()` implemented to allow optimization. The `retract()` interface adds an increment in local coordinates to the line, and contains logic to always return a parametrization with a positive range ρ.

The line factor introduced here, together with range-bearing factors for point measurements and relative pose factors for the odometry form are used to form a factor graph which is locally consistent around the robot. The result can be seen in Figure 3.2, where a number of lines, corners and circles have been measured multiple times and given unique `id`'s on the local map. The measurements taken from a certain robot pose are shown as grey lines.

A.1.2 Improving odometry with local ICP

To improve local odometry in the presence of drift and time synchronization inaccuracies, a local *Iterative Closest Point* matching is used to obtain improved odometry by using two consecutive laser scans. The rationale behind this is that the *true odometry* is the one matching two scans perfectly, which is slightly different from the encoder odometry due to drift, timing mismatch and mounting position error. It must be noted, however, that ICP in environments where the static assumption is violated may possibly yield worse results than exclusively using odometry.

A

Appendix B

Offline Pose Estimation

B.1 Pose Estimation

In this appendix we explain the tracking mode that was used to obtain the ground truth trajectories against the accurate map of the building. In tracking mode, the local graph as explained in Chapter Three is related to the global frame by making nearest-neighbor data associations with the global map, after an initial pose estimate has been provided by the user. These associations are then incorporated in the graph as location priors for the local features expressed in the global map reference frame, resulting in robot poses that are also valid with respect to the global map. The tracking mode can be started in our implementation by providing an initial pose estimate in RVIZ. For our ground truth, we manually supplied the starting position of the robot for each trajectory and verified the data associations that were made. The resulting robot poses at sample intervals were then used in table 3.3 as ground truth.

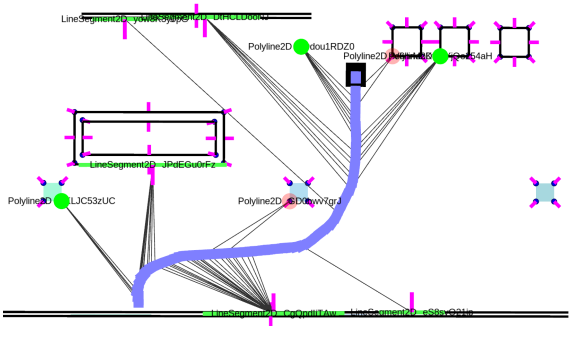


Figure B.1. *The local map of features (green) aligned against the global map, which serves as a prior using nearest neighbor data associations. The resulting robot trajectory is the MAP estimate of the robot poses in the global map. We used this method to obtain the ground truth estimates for our trajectories to verify our hypothesis approach. The robot trajectory is indicated by the blue line and the green and red circles and lines indicate features.*

B

Appendix C

Map Representation and Tooling

This appendix provides an overview of the concepts and software that are used to model the maps in this thesis. The maps from the Atlas and Gemini environment are represented in the JSON-LD [72] host language. This representation is then stored in a PostgreSQL [104] database. The localization executable is written in C++ and uses libpqxx [85] to perform SQL queries. We used the 3D modeling software Blender [11] to edit geometry for the world models and export JSON-LD by using the python scripting capabilities. Some of the modeling decisions, advantages and disadvantages are discussed in this appendix.

C.1 JSON-LD Models

JSON-LD is an open source standard for representing linked (i.e., graph) data that comes with an Application Programming Interface (API) that allows to perform basic queries. An example of JSON-LD is shown below:


```
{
    "@id": "ColumnCircular.001",
    "@type": "Column",
    "represented_by": [{
        "@id": "3e1e0df0",
        "@type": "ObjectFeatureRepresentation",
        "perceivable_by": {
            "@id": "PlanarLidar2D"
        },
        "represented_by": {
            "@type": "Circle",
            "@id": "04ed3677",
            "radius": 0.33,
            "has_placement": {
                "@id": "8a900c1a",
                "@type": "Point",
            }
        }
```

```
    }]
}
```

As can be seen, JSON-LD extends JSON objects with an `@id` and `@type` specifier. These symbolic identifiers can be used to link to other objects in the same document, or to resources outside the document using internationalized resource identifiers (IRI). For example, the `Point` object that is referenced by the `Circle` object can be defined separately in the document:

```
{
    "@id": "8a900c1a",
    "@type": "Point",
    "has_position": {
        "@type": "Position",
        "@id": "0286df6a",
        "coordinates": [
        {
            "@type": "CartCoord2D",
            "x": 0.0,
            "y": 6.11254358291626
        }
        ]
    }
}
```

This referencing capability enables both nested definitions that can be linked to from outside the nesting scope, as well as *flattened* definitions where no objects are nested at all. With this graph structure we separate geometric definitions from their relations to objects, sensors and patterns. Furthermore, we can maintain topological consistency in the definition of geometric entities. For example, a `Point` model can be used as part of both a wall, a corner, and a polygon that is used for navigation. The PyLD [52] Python API is used to extract relevant objects from the models by using *frames*.

C.1.1 Framing queries

C

The JSON-LD API provides *framing*, which is used to extract objects of a certain type from a document, while also nesting relations in that object according to a specified hierarchy. An example of this is when we are interested in all geometric objects (such as polylines) in order to visualize them, together with the objects that they represent. A frame can be used to query for these objects:

```
{
"@context":{
    "dummy://navigation/represents":{"@reverse":"dummy://navigation/↩
        represented_by"}
},
"@type":"dummy://point_geometry_2d/polyline",
"@explicit": false,
"dummy://navigation/represents": {},
"dummy://point_geometry_2d/enclosing": {}
}
```

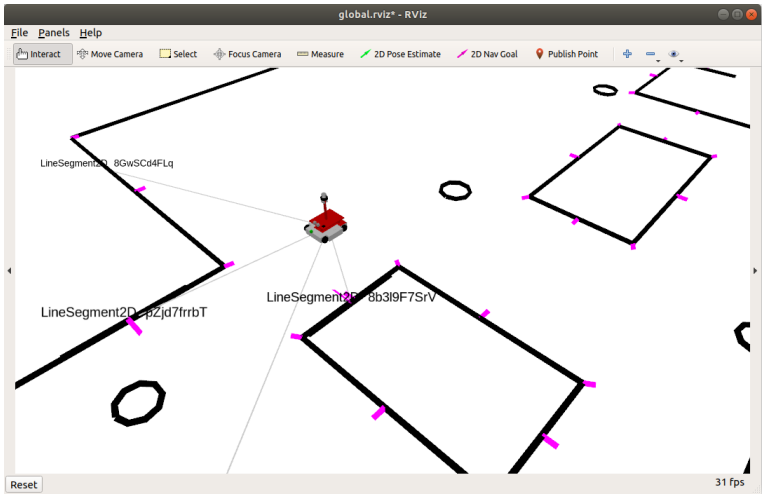


Figure C.1. *The RVIZ visualization of a robot model and the geometry of the world*

This frame specifies that we want a list of `Polyline` objects from the document, with nested relations that specify the object they represent and their enclosing (i.e., on which side they enclose a volume if they do so). Note that an `@context` specifier is added in which a reverse relation is defined. This allows the API to find relations that are specified from subject to object and embed the reverse relation in the object. IRIs can be aliased in documents to decrease the file size, but in this frame the full IRIs are used.

C.1.2 Visualization

For visualization, the well-known RVIZ program is used which is part of ROS. The geometric objects are queried from a JSON-LD document by the API and are published in the map reference frame, allowing them to be overlayed with sensor data and a robot model. This is shown in Figure C.1

C.2 PostgreSQL and PostGIS

Data from the JSON-LD graph is stored in a PostgreSQL database, from which it is queried by the localization and motion executables. The benefit of this approach over directly importing the JSON data is that (spatial) SQL queries are used to access the data and (spatial) indices are supported. The latter was not critical in the current work as the maps and features we use resulted in manageable table sizes. The JSON-LD nodes and relations are directly inserted into database tables and a separate table is used for the PostGIS geometry entities. The PostGIS geometry type is a special binary type that follows the well-known-text specification

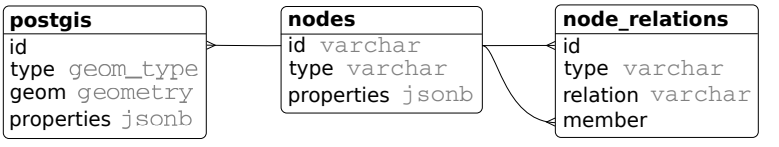


Figure C.2. *The SQL tables used to store the JSON-LD. The JSON-LD graph is stored in the* nodes *and* node_relations *table. The* postgis *table contains a copy of the spatial objects in the graph.*

of geometry. This specification does not consider topology (i.e., the sharing of points between multiple geometries) and we only use it as a redundant table on top of the geometries that were already defined as graph nodes, for spatial queries. The geometry type only supports geographically referenced coordinate systems, so a dummy Cartesian reference system is used to which we attribute no geographic meaning. A benefit of this approach is that geometry can also be visualized with PostGIS visualization tools. The schema for the database is shown in Figure C.2.

C.2.1 Queries

Spatial queries were the main reason we chose PostGIS as a database system. By a combination of spatial queries and `JOIN` statements, queries can be performed that, e.g., request geometry inside a certain region that represents, for instance, a door or a wall. An example of such a query is shown below:

```
with

OFR as (select id from node_relations where type='dummy://navigation/objectfeaturerepresentation' and member='↩
      PlanarLidar2D'),

OFRandGeom as ( select OFR.id, node_relations.member as geoid from
OFR inner join node_relations on OFR.id=node_relations.id where node_relations.relation='dummy://navigation/↩
      represented_by' ),

OFRandPostgis as
(select OFRandGeom.id as OFRid, postgis.id as postgisid, postgis.type as postgistype, ST_ASTEXT(postgis.geom) as ↩
      geomtext,
geom, st_distance(st_geomfromtext( 'POINT(0 0)',3035),postgis.geom) as distance from
OFRandGeom inner join postgis on OFRandGeom.geoid=postgis.id and st_distance(st_geomfromtext( 'POINT(0 0)',3035),↩
      postgis.geom) < 10 order by distance),

OFRandPostgisandProps as
(select OFRid, postgisid, postgistype, geomtext, geom, distance, nodes.properties from
OFRandPostgis inner join nodes on nodes.id = postgisid )

select distinct on (node_relations.id) node_relations.type,  node_relations.id, OFRandPostgisandProps.postgistype, ↩
       OFRandPostgisandProps.postgisid, geomtext, properties,  geom, distance from
(node_relations join OFRandPostgisandProps on OFRandPostgisandProps.OFRid=node_relations.member and node_relations.↩
      relation='dummy://navigation/represented_by'
and node_relations.member = OFRid and node_relations.type='dummy://navigation/door')
order by  node_relations.id, distance;
```

This query selects all doors and walls with their properties and geometry representations within a 10 meter radius of a reference point. The output is:

	type character varying	Id character varying	postgistype... entry_type	postgisId character varyin	geomtext text	properties jsonb	geom geometry	distance double precision
1	dummy://navigation/door	IfcDoor/door bathroom:door bathroom:512...	Polygon	961f9284	POLYGON((7.08...	{}	0103000020DB0B...	9.93291913126597
2	dummy://navigation/door	IfcDoor/Simple door:Simple door:512765	Polygon	e15b1550	POLYGON((6.88...	{}	0103000020DB0B...	7.90966966309795
3	dummy://navigation/wall	IfcWallStandardCase/Basic Wall:Internal w...	Polyline	8923296d	LINESTRING(8.0...	{"dummy://p...	0102000020DB0B...	8.78231361759041
4	dummy://navigation/wall	IfcWallStandardCase/Basic Wall:Internal w...	Polyline	3cd5c0e4	LINESTRING(6.7...	{"dummy://p...	0102000020DB0B...	7.56196880449904
5	dummy://navigation/wall	IfcWallStandardCase/Basic Wall:Internal w...	Polyline	c165da50	LINESTRING(6.7...	{"dummy://p...	0102000020DB0B...	7.55912590397086
6	dummy://navigation/wall	IfcWallStandardCase/Basic Wall:Internal w...	Polyline	3038f73c	LINESTRING(6.8...	{"dummy://p...	0102000020DB0B...	9.72179779910095

From this query it can be seen that while the spatial capabilities are useful, declaring the necessary graph traversals is quite cumbersome in the SQL language. Better alternatives in the form of graph databases and graph query languages provide more promising alternatives in this domain (see e.g., [99]). Although we are not aware of any spatial extensions for graph databases that are similarly mature as the PostGIS extension. For the multi-hypothesis localization implementation, the database is only queried when the executable starts. The evaluation of feature candidates is very domain-specific and requires high performance which can only be achieved application-side.

C.3 Blender

Blender is a modeling tool that is used for creating, animating and rendering static scenes and movies. It has already been applied in the robotics domain for visualization and simulation [19]. In this thesis, it used for the creation of maps that are exported to JSON-LD representations using the Python scripting interface.

C.3.1 Modeling

The modeling of maps is done by first importing a grid map as a reference (e.g., build by a SLAM method), and then placing geometric objects on top of this map that resemble the walls, columns and doors. These objects are three dimensional, but only the vertices with height $z = 0$ are extracted for every object to generate the 2D representation for the planar LiDAR. The object type is inferred from the name of each object (e.g. a circular column must start with "CircularColumn"). A python script loops over all objects and extracts the corner vertices which are then added as geometric representations to the objects that are linked to a planar LiDAR.

C.3.2 BIM models

A Blender plugin that is maintained by the BIM community [12] allows to import IFC models into Blender. It is based on the IfcOpenShell library that can convert the geometric entities to mesh representations. The plugin provides an alternative to the conversion path in Chapter Two for converting BIM data from the robotics domain. It can provide a useful addition to the workflow, especially for exporting and working with 3D data in future work.

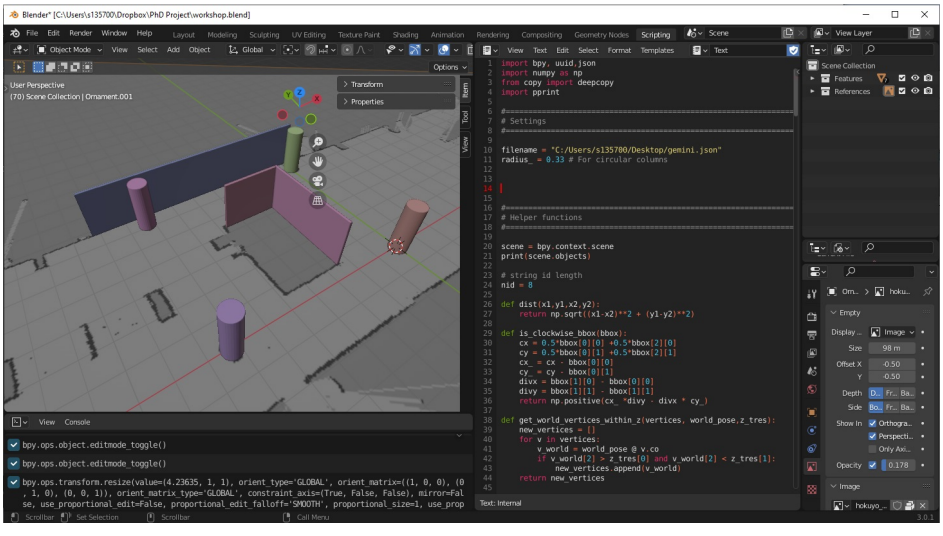


Figure C.3. *Modeling an indoor environment in Blender on top of grid map that is obtained by performing SLAM.*

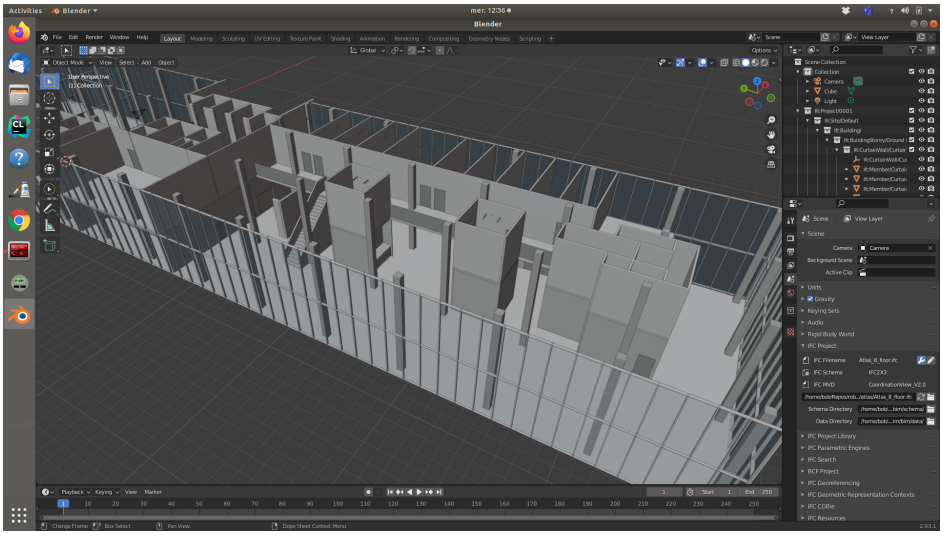


Figure C.4. *using the BlenderBIM plugin to import BIM geometry [12]*

C

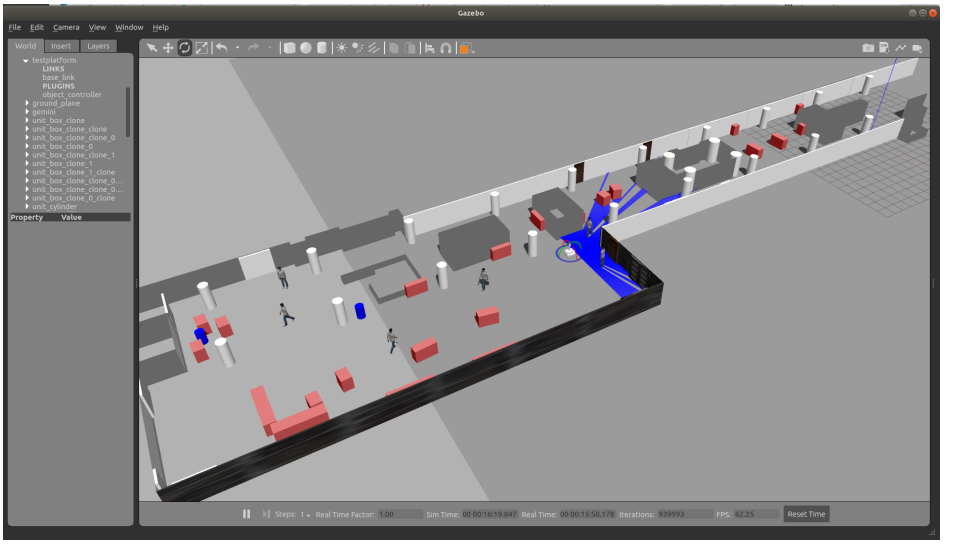


Figure C.5. *A Gazebo simulation that is used to test the localization algorithm.*

C.3.3 Gazebo simulation

To test the localization in this thesis, we exported models from Blender and imported them into the Gazebo simulation environment [50]. An example is shown in Figure C.5, where objects are added to the static geometry and a model of the reference platform is used in a closed loop simulation.

Bibliography

[1] D. Acharya, K. Khoshelham, and S. Winter, "BIM-PoseNet: Indoor camera localisation using a 3D indoor model and deep learning from synthetic images", *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 150, pp. 245–258, 2019.

[2] R. Almadhoun, T. Taha, L. Seneviratne, J. Dias, and G. Cai, "A survey on inspecting structures using robotic systems", *International Journal of Advanced Robotic Systems*, vol. 13, no. 6, pp. 1–18, 2016.

[3] C. Alves, A. Cardoso, A. Colim, E. Bicho, A. C. Braga, J. Cunha, C. Faria, and L. A. Rocha, "Human–Robot Interaction in Industrial Settings: Perception of Multiple Participants at a Crossroad Intersection Scenario with Different Courtesy Cues", *Robotics*, vol. 11, no. 3, p. 59, 2022.

[4] K. O. Arras, J. A. Castellanos, M. Schilt, and R. Siegwart, "Feature-based multi-hypothesis localization and tracking using geometric constraints", *Robotics and Autonomous Systems*, vol. 44, pp. 41–53, 2003.

[5] *Autonomous Mobile Robots Market Size, Fortune | Report [2028]*. [Online]. Available: `https://www.fortunebusinessinsights.com/autonomous-mobile-robots-market-105055` (visited on 07/28/2022).

[6] O. Aycard and C. Brouard, "A new tool to initialize global localization for a mobile robot", in *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*, 2020, pp. 1296–1303. [Online]. Available: `https://hal.archives-ouvertes.fr/hal-03007286`.

[7] T. Barros, R. Pereira, L. Garrote, C. Premebida, and U. J. Nunes, "Place recognition survey: An update on deep learning approaches", *arXiv*, 2021. [Online]. Available: `http://arxiv.org/abs/2106.10458`.

[8] H. Bavle, J. L. Sanchez-Lopez, M. Shaheer, J. Civera, and H. Voos, "Situational Graphs for Robot Navigation in Structured Indoor Environments", *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9107–9114, 2022.

[9] J. Berg, A. Lottermoser, C. Richter, and G. Reinhart, "Human-Robot-Interaction for mobile industrial robot teams", in *Procedia CIRP*, vol. 79, Elsevier, Jan. 2019, pp. 614–619.

[10] P. Bhattacharya and M. Gavrilova, "A survey of landmark recognition using the bag-of-words framework", in *Studies in Computational Intelligence*, vol. 441, New York: Springer, 2013, pp. 243–263.

[11] *blender.org - Home of the Blender project - Free and Open 3D Creation Software*. [Online]. Available: `https://www.blender.org/` (visited on 09/23/2022).

[12] *BlenderBIM Add-on*. [Online]. Available: `https://blenderbim.org/` (visited on 09/23/2022).

[13] F. Boniardi, T. Caselitz, R. Kummerle, and W. Burgard, "Robust LiDAR-based localization in architectural floor plans", in *IEEE International Conference on Intelligent Robots and Systems*, 2017, pp. 3318–3324.

[14] F. Boniardi, T. Caselitz, R. Kümmerle, and W. Burgard, "A pose graph-based localization system for long-term navigation in CAD floor plans", *Robotics and Autonomous Systems*, vol. 112, pp. 84–97, 2019.

[15] *Boost geometry - buffer*. [Online]. Available: `https://www.boost.org/doc/libs/1_80_0/libs/geometry/doc/html/geometry/reference/algorithms/buffer/buffer_7_with_strategies.html` (visited on 08/15/2022).

[16] A. Borrmann, M. König, C. Koch, and J. Beetz, "Building Information Modeling: Why? What? How?", in *Building Information Modeling: Technology Foundations and Industry Practice*. New York: Springer, 2018, ch. 1, pp. 1–24.

[17] D. Brugali, P. Scandurra, and R. B. Blocks, "Component-based Robotic Engineering Part I: Reusable building blocks", *Robotics Automation Magazine*, vol. 16, no. 4, pp. 84–96, 2009.

[18] H. Bruyninckx, *Building blocks for complicated and situational aware robotic and cyber-physical systems*. [Online]. Available: `https://robmosys.pages.gitlab.kuleuven.be/` (visited on 07/20/2022).

[19] K. Buys, T. De Laet, R. Smits, and H. Bruyninckx, "Blender for robotics: Integration into the Leuven paradigm for robot task specification and human motion estimation", in *Lecture Notes in Computer Science (Artificial Intelligence)*, vol. 6472 LNAI, Springer, 2010, pp. 15–25.

[20] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age", *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[21] H. Chen, R. Hendrikx, M. J. G. Van De Molengraft, and H. Bruyninckx, "Behavior adaptation for mobile robots via semantic map compositions of constraint-based controllers", *(Submitted for publication)*, 2022.

[22] S. Y. Chen, "Kalman filter for robot vision: A survey", *IEEE Transactions on Industrial Electronics*, vol. 59, no. 11, pp. 4409–4420, 2012.

[23] X. Chen, Y. Chen, and J. Chase, “Mobiles Robots - Past Present and Future”, in *Mobile Robots - State of the Art in Land, Sea, Air, and Collaborative Missions*, London: IntechOpen, 2009.

[24] G. Ciaparrone, F. Luque Sánchez, S. Tabik, L. Troiano, R. Tagliaferri, and F. Herrera, “Deep learning in video multi-object tracking: A survey”, *Neurocomputing*, vol. 381, pp. 61–88, 2020.

[25] J. Crespo, J. C. Castillo, O. M. Mozos, and R. Barber, “Semantic information for robot navigation: A survey”, *Applied Sciences*, vol. 10, no. 2, 2020.

[26] L. Custodio and R. Machado, “Flexible automated warehouse: a literature review and an innovative framework”, *International Journal of Advanced Manufacturing Technology*, vol. 106, no. 1-2, pp. 533–558, 2020.

[27] K. Dautenhahn, A. Campbell, and D. S. Syrdal, “Does anyone want to talk to me? - Reflections on the use of assistance and companion robots in care homes”, in *Procs 4th Int Symposium on New Frontiers in Human-Robot Interaction*, 2015.

[28] S. J. Davey, “Simultaneous localization and map building using the probabilistic multi-hypothesis tracker”, *IEEE Transactions on Robotics*, vol. 23, no. 2, pp. 271–280, 2007.

[29] R. De Koster, “Automated and Robotic Warehouses: Developments and Research Opportunities”, *Logistics and Transport*, vol. 38, no. 2, 2018.

[30] R. de Koning, E. Torta, P. Pauwels, R. W. M. Hendrikx, and M. J. G. Van De Molengraft, “Queries on Semantic Building Digital Twins for Robot Navigation Citation for published version (APA): de Koning Queries on Semantic Building Digital Twins for Robot Navigation”, in *9th Linked Data in Architecture and Construction Workshop*, 2021, pp. 32–42.

[31] H. Deeken, T. Wiemann, and J. Hertzberg, “Grounding semantic maps in spatial databases”, *Robotics and Autonomous Systems*, vol. 105, pp. 146–165, 2018.

[32] F. Dellaert, “Factor Graphs and GTSAM”, Tech. Rep. GT-RIM-CP&R-2012-002, 2012, pp. 1–27. [Online]. Available: `http://tinyurl.com/gtsam`.

[33] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, “Monte Carlo localization for mobile robots”, *International Conference on Robotics and Automation*, vol. 2, pp. 1322–1328, 1999.

[34] D. H. Dos Reis, D. Welfer, M. A. De Souza Leite Cuadros, and D. F. T. Gamarra, “Mobile Robot Navigation Using an Object Recognition Software with RGBD Images and the YOLO Algorithm”, *Applied Artificial Intelligence*, vol. 33, no. 14, pp. 1290–1305, 2019.

[35] C. Eastman, P. Teicholz, R. Sacks, and K. Liston, *BIM handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors*, 2nd. Hoboken, New York, United States: John Wiley & Sons, 2011, p. 650.

[36] M. Eder, M. Reip, and G. Steinbauer, "Creating a robot localization monitor using particle filter and machine learning approaches", *Applied Intelligence*, vol. 52, no. 6, pp. 6955–6969, 2022.

[37] A. Elfes, "Using Occupancy Grids for Mobile Robot Perception and Navigation", *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[38] J. Elfring, E. Torta, and R. van de Molengraft, "Particle filters: A hands-on tutorial", *Sensors*, vol. 21, no. 2, pp. 1–28, 2021.

[39] *ExRobotics: The robot operator for unmanned facilities | ExRobotics*. [Online]. Available: `https://exrobotics.global/` (visited on 08/26/2022).

[40] J. P. Fentanes, B. Lacerda, T. Krajnik, N. Hawes, and M. Hanheide, "Now or later? Predicting and maximising success of navigation actions from long-term experience", in *International Conference on Robotics and Automation*, IEEE, 2015, pp. 1112–1117.

[41] M. A. Fischler and R. C. Bolles, "Random sample consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography", *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[42] D. Fox, "Adapting the sample size in particle filters through KLD-sampling", *International Journal of Robotics Research*, vol. 22, no. 12, pp. 985–1003, 2003.

[43] D. Fox, W. Burgard, and S. Thrun, "Active Markov localization for mobile robots", *Robotics and Autonomous Systems*, vol. 25, no. 3-4, pp. 195–207, 1998.

[44] D. Fox, J. Hightower, L. Liao, D. Schulz, and G. Bordello, "Bayesian filtering for location estimation", *IEEE Pervasive Computing*, vol. 2, no. 3, pp. 24–33, 2003.

[45] J. Fuegi and J. Francis, "Lovelace & Babbage and the Creation of the 1843 'Notes'", *IEEE Annals of the History of Computing*, vol. 25, no. 4, pp. 16–26, 2003.

[46] C. Galindo, J.-A. Fernández-Madrigal, J. González, and A. Saffiotti, "Robot task planning using semantic maps", *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 955–966, 2008.

[47] Gallant, *GitHub - kam3k/laser-line-extraction: A ROS package that extracts line segments from LaserScan messages.* 2014. [Online]. Available: `https://github.com/kam3k/laser_line_extraction` (visited on 09/29/2020).

[48] D. Gálvez-López and J. D. Tardós, "Bags of binary words for fast place recognition in image sequences", *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.

[49] H. Gao, X. Zhang, J. Yuan, J. Song, and Y. Fang, "A Novel Global Localization Approach Based on Structural Unit Encoding and Multiple Hypothesis Tracking", *IEEE Transactions on Instrumentation and Measurement*, vol. 68, no. 11, pp. 4427–4442, 2019.

[50] *Gazebo*. [Online]. Available: `https://gazebosim.org/home` (visited on 09/26/2022).

[51] *GitHub - BehaviorTree/BehaviorTree.CPP: Behavior Trees Library in C++*. [Online]. Available: `https://github.com/BehaviorTree/BehaviorTree.CPP` (visited on 08/15/2022).

[52] *GitHub - digitalbazaar/pyld: JSON-LD processor written in Python*. [Online]. Available: `https://github.com/digitalbazaar/pyld` (visited on 09/23/2022).

[53] C. Gomez, A. C. Hernandez, J. Crespo, and R. Barber, "A topological navigation system for indoor environments based on perception events", *International Journal of Advanced Robotic Systems*, vol. 14, no. 1, 2016.

[54] I. Ha, H. Kim, S. Park, and H. Kim, "Image retrieval using BIM and features from pretrained VGG network for indoor localization", *Building and Environment*, vol. 140, pp. 23–31, 2018.

[55] M. Hanheide, M. Göbelbecker, G. S. Horn, A. Pronobis, K. Sjöö, A. Aydemir, P. Jensfelt, C. Gretton, R. Dearden, M. Janicek, H. Zender, G. J. Kruijff, N. Hawes, and J. L. Wyatt, "Robot task planning and explanation in open and uncertain worlds", *Artificial Intelligence*, vol. 247, pp. 119–150, 2017.

[56] T. He and S. Hirose, "A global localization approach based on Line-segment Relation Matching technique", *Robotics and Autonomous Systems*, vol. 60, no. 1, pp. 95–112, 2012.

[57] R. W. M. Hendrikx, H. Bruyninckx, J. Elfring, and M. J. G. Van De Molengraft, "Local-To-Global Hypotheses for Robust Robot Localization", *Frontiers in Robotics and AI*, 2022.

[58] R. W. M. Hendrikx, H. Bruyninckx, H. L. Chen, J. Elfring, and M. J. G. Van De Molengraft, "From features to object patterns for indoor robot localization", *(Submitted for publication)*, 2023.

[59] R. W. M. Hendrikx, P. Pauwels, E. Torta, H. J. Bruyninckx, and M. J. G. van de Molengraft, "Connecting Semantic Building Information Models and Robotics: An application to 2D LiDAR-based localization", in *Proceedings - IEEE International Conference on Robotics and Automation*, 2021, pp. 11 654–11 660.

[60] M. Himstedt, J. Frost, S. Hellbach, H. J. Bohme, and E. Maehle, "Large scale place recognition in 2D LIDAR scans using Geometrical Landmark Relations", *IEEE International Conference on Intelligent Robots and Systems*, pp. 5030–5035, 2014.

[61] M. Himstedt and E. Maehle, "Semantic Monte-Carlo localization in changing environments using RGB-D cameras", in *2017 European Conference on Mobile Robots*, 2017, pp. 1–8.

[62] *Holland High Tech*. [Online]. Available: `https://hollandhightech.nl/` (visited on 09/07/2022).

[63] D. Hong, G. Heo, C. W. Myung, and W. S. Ra, "Data association approach to mobile robot localization using non-unique landmarks", *International Conference on Control, Automation and Systems*, pp. 1049–1053, 2017.

[64] *IFCJSON-Team/IFC.JSON-4: Repository containing the specification for IFC.JSON*. [Online]. Available: `https://github.com/IFCJSON-Team/IFC.JSON-4` (visited on 08/19/2020).

[65] *Industry Foundation Classes (IFC) - buildingSMART Technical*. [Online]. Available: `https://technical.buildingsmart.org/standards/ifc/` (visited on 08/19/2020).

[66] F. Ingrand and M. Ghallab, "Deliberation for autonomous robots: A survey", *Artificial Intelligence*, vol. 247, pp. 10–44, 2017.

[67] *iRobot®: Robot Vacuums and Mops*. [Online]. Available: `https://www.irobot.com/` (visited on 07/28/2022).

[68] P. Jensfelt and S. Kristensen, "Active global localization for a mobile robot using multiple hypothesis tracking", *IEEE Transactions on Robotics and Automation*, vol. 17, no. 5, pp. 748–760, 2001.

[69] J. Jessup, S. N. Givigi, and A. Beaulieu, "Robust and efficient multi-robot 3D mapping with octree based occupancy grids", in *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, IEEE, 2014, pp. 3996–4001.

[70] E. Johns and G. Z. Yang, "Global localization in a dense continuous topological map", in *International Conference on Robotics and Automation*, 2011, pp. 1032–1037.

[71] C. E. Johnson, "Topological Mapping and Navigation in Real-World Environments", Dissertation, University of Michigan, 2018.

[72] *JSON-LD - JSON for Linking Data*. [Online]. Available: `https://json-ld.org/` (visited on 09/23/2022).

[73] *JSON-LD 1.1 Framing*. [Online]. Available: `https://json-ld.org/spec/latest/json-ld-framing/` (visited on 08/19/2020).

[74] H. K. Kang and K. J. Li, "A standard indoor spatial data model - OGC IndoorGML and implementation approaches", *ISPRS International Journal of Geo-Information*, vol. 6, no. 4, p. 116, 2017.

[75] S. Kaszuba, S. R. Sabbella, V. Suriani, F. Riccio, and D. Nardi, "RoSmEEry: Robotic Simulated Environment for Evaluation and Benchmarking of Semantic Mapping Algorithms", *arXiv preprint*, 2021.

[76] K. Khalvati and A. K. MacKworth, “Active robot localization with macro actions”, in *International Conference on Intelligent Robots and Systems*, 2012, pp. 187–193.

[77] I. Kostavelis and A. Gasteratos, “Semantic mapping for mobile robotics tasks: A survey”, *Robotics and Autonomous Systems*, vol. 66, pp. 86–103, 2015.

[78] R. Kuemmerle, M. Ruhnke, B. Steder, C. Stachniss, and W. Burgard, “Autonomous Robot Navigation in Highly Populated Pedestrian Zones”, *Journal of Field Robotics*, vol. 32, no. 4, pp. 565–589, 2015.

[79] B. Kuipers, J. Modayil, P. Beeson, M. Macmahon, and F. Savelli, “Local Metrical and Global Topological Maps in the Hybrid Spatial Semantic Hierarchy”, in *International Conference on Robotics and Automation*, IEEE, 2004.

[80] L. Kunze, N. Hawes, T. Duckett, M. Hanheide, and T. Krajnik, “Artificial Intelligence for Long-Term Robot Autonomy: A Survey”, *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4023–4030, 2018.

[81] Y. Latif, Ć. Cadena, and J. Neira, “Robust graph SLAM back-ends: A comparative analysis”, in *IEEE International Conference on Intelligent Robots and Systems*, IEEE, 2014, pp. 2683–2690.

[82] *Lely Juno*. [Online]. Available: `https://www.lely.com/nl/oplossingen/voeren/juno/` (visited on 03/09/2023).

[83] K. J. Li, S. Zlatanova, J. Torres-Sospedra, A. Perez-Navarro, C. Laoudias, and A. Moreira, “Survey on indoor map standards and formats”, in *2019 International Conference on Indoor Positioning and Indoor Navigation*, IEEE, 2019.

[84] Y. Li and E. B. Olson, “IPJC: The incremental posterior joint compatibility test for fast feature cloud matching”, in *IEEE International Conference on Intelligent Robots and Systems*, 2012, pp. 3467–3474.

[85] *libpqxx: the official C++ language binding for PostgreSQL*. [Online]. Available: `http://pqxx.org/development/libpqxx/` (visited on 09/23/2022).

[86] F. Lu and E. Milios, “Globally Consistent Range Scan Alignment for Environment Mapping”, *Autonomous Robots*, vol. 4, no. 4, pp. 333–349, 1997.

[87] M. Lutz, S. Hochdorfer, and C. Schlegel, “Global localization using multiple hypothesis tracking: A real-world approach”, *2011 IEEE Conference on Technologies for Practical Robot Applications*, pp. 127–132, 2011.

[88] 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 *International Conference on Intelligent Robots and Systems (IROS)*, IEEE/RSJ, 2020, pp. 2–7.

[89] M. Montemerlo, S. Thrun, and W. Whittaker, "Conditional particle filters for simultaneous mobile robot localization and people-tracking", in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 1, 2002, pp. 695–701.

[90] T. Morris, F. Dayoub, P. Corke, and B. Upcroft, "Simultaneous localization and planning on multiple map hypotheses", *IEEE International Conference on Intelligent Robots and Systems*, pp. 4531–4536, 2014.

[91] A. C. Murtra, J. M. Mirats Tur, and A. Sanfeliu, "Efficient active global localization for mobile robots operating in large and cooperative environments", *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2758–2763, 2008.

[92] L. Naik, S. Blumenthal, N. Huebel, H. Bruyninckx, and E. Prassler, "Semantic mapping extension for OpenStreetMap applied to indoor robot navigation", *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May, pp. 3839–3845, 2019.

[93] M. M. Neggers, R. H. Cuijpers, P. A. Ruijten, and W. A. IJsselsteijn, "Determining Shape and Size of Personal Space of a Human when Passed by a Robot", *International Journal of Social Robotics*, vol. 14, no. 2, pp. 561–572, 2022.

[94] J. Neira and J. D. Tardós, "Data association in stochastic mapping using the joint compatibility test", *IEEE Transactions on Robotics and Automation*, vol. 17, no. 6, pp. 890–897, Dec. 2001.

[95] N. J. Nilsson, "Shakey the robot", SRI International, Tech. Rep. 323, 1984.

[96] E. Olson and P. Agarwal, "Inference on networks of mixtures for robust robot mapping", in *Robotics: Science and Systems*, vol. 8, 2013, pp. 313–320.

[97] W. Palacz, G. Ślusarczyk, B. Strug, and E. Grabska, "Indoor robot navigation using graph models based on BIM/IFC", in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11509 LNAI, Springer Verlag, 2019, pp. 654–665.

[98] D. Paulius and Y. Sun, "A Survey of Knowledge Representation in Service Robotics", *Robotics and Autonomous Systems*, vol. 118, pp. 13–30, Aug. 2019.

[99] P. Pauwels, A. Van Duijven, R. De Koning, R. Hendrikx, and E. Torta, "Semantic Data from Building Digital Twins for Robot Navigation: File-Based and RealTime Data Transfer Methods", *(Submitted for publication)*, 2022.

[100] P. Pauwels and W. Terkaj, "EXPRESS to OWL for Construction Industry: Towards a Recommendable and Usable ifcOWL Ontology", *Automation in Construction*, vol. 63, pp. 100–133, 2016.

[101] T. Pfeifer, P. Weissig, S. Lange, and P. Protzel, "Robust factor graph optimization - A comparison for sensor fusion applications", in *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, IEEE, 2016.

[102] S. T. Pfister, S. I. Roumeliotis, and J. W. Burdick, "Weighted line fitting algorithms for mobile robot map building and efficient data representation", in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 1, 2003, pp. 1304–1311.

[103] F. Pomerleau, F. Colas, and R. Siegwart, "A Review of Point Cloud Registration Algorithms for Mobile Robotics A Review of Point Cloud Registration Al-gorithms for Mobile Robotics", *Foundations and Trends in Robotics*, vol. 4, no. 1, pp. 1–104, 2015.

[104] *PostgreSQL: The world's most advanced open source database*. [Online]. Available: `https://www.postgresql.org/` (visited on 09/23/2022).

[105] A. Pronobis and P. Jensfelt, "Large-scale Semantic Mapping and Reasoning with Heterogeneous Modalities", in *International Conference on Robotics and Automation*, IEEE, 2012, pp. 3515–3522.

[106] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System", in *Workshops at the IEEE International Conference on Robotics and Automation*, 2009.

[107] K. Rajan and A. Saffiotti, "Towards a science of integrated AI and Robotics", *Artificial Intelligence*, vol. 247, pp. 1–9, 2017.

[108] J. Reuter, "Scan- and featurebased multiple hypothesis tracking for mobile robot localization: a data fusion approach", *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 4, pp. 714–719, 1999.

[109] *Ropod*, 2020. [Online]. Available: `https://cordis.europa.eu/project/id/731848` (visited on 01/03/2022).

[110] N. Roy, G. Gordon, and S. Thrun, "Finding Approximate POMDP Solutions Through Belief Compression", *Journal of Artificial Intelligence Research*, vol. 23, pp. 1–40, 2005.

[111] F. Rubio, F. Valero, and C. Llopis-Albert, "A review of mobile robots: Concepts, methods, theoretical framework, and applications", *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, 2019.

[112] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF", in *Proceedings of the IEEE International Conference on Computer Vision*, 2011, pp. 2564–2571.

[113] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison, "SLAM++: Simultaneous localisation and mapping at the level of objects", in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1352–1359.

[114] C. Schlette and J. Roßmann, "Sampling-Based Floor Plan Analysis on BIMs", in *Proceedings of the 33rd International Symposium on Automation and Robotics in Construction (ISARC)*, International Association for Automation and Robotics in Construction (IAARC), 2016, pp. 28–35.

[115] T. Serre, "Deep Learning: The Good, the Bad, and the Ugly", *Annual Review of Vision Science*, vol. 5, pp. 399–426, 2019.

[116] M. Shahzad, M. T. Shafiq, D. Douglas, and M. Kassem, "Digital Twins in Built Environments: An Investigation of the Characteristics, Applications, and Challenges", *Buildings*, vol. 12, no. 2, p. 120, 2022.

[117] X. Shen, E. Frazzoli, D. Rus, and M. H. Ang, "Fast Joint Compatibility branch and bound for feature cloud matching", in *IEEE International Conference on Intelligent Robots and Systems*, IEEE, 2016, pp. 1757–1764.

[118] T. Shimmura, R. Ichikari, T. Okuma, H. Ito, K. Okada, and T. Nonaka, "Service robot introduction to a restaurant enhances both labor productivity and service quality", in *Procedia CIRP*, vol. 88, Elsevier, 2020, pp. 589–594.

[119] R. Simmons and S. Koenig, "Probabilistic Robot Navigation in Partially Observable Environments", *Proceedings of the 1995 International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1080–1087, 1995.

[120] *Simple Feature Access - Part 1: Common Architecture | OGC*. [Online]. Available: `https://www.ogc.org/standards/sfa` (visited on 09/28/2020).

[121] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics", in *IEEE International Conference on Robotics and Automation*, IEEE, 1987.

[122] N. Sunderhauf and P. Protzel, "Switchable constraints vs. max-mixture models vs. RRR - A comparison of three approaches to robust pose graph SLAM", in *Proceedings - IEEE International Conference on Robotics and Automation*, 2013, pp. 5198–5203.

[123] A. Tabrez, M. B. Luebbers, and B. Hayes, "A Survey of Mental Modeling Techniques in Human–Robot Teaming", *Current Robotics Reports*, vol. 1, pp. 259–267, 2020.

[124] *Taski*. [Online]. Available: `https://taski.com/` (visited on 03/09/2023).

[125] M. Tenorth and M. Beetz, "Representations for robot knowledge in the KNOWROB framework", *Artificial Intelligence*, vol. 247, pp. 151–169, 2017.

[126] S. U. Tessema LS Jaeger R, "Extraction of an IndoorGML Model from an Occupancy Grid Map Constructed using 2D LiDAR", *39. Wissenschaftlich-Technische Jahrestagung der DGPF*, vol. 28, pp. 97–110, 2019.

[127] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[128] S. Thrun and M. Montemerlo, "The graph SLAM algorithm with applications to large-scale mapping of urban structures", *International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 403–429, 2006.

[129] M. F. B. Van Der Burgh, J. J. M. Lunenburg, R. P. W. Appeldoorn, L. L. A. M. Van Beek, J. Geijsberts, L. G. L. Janssen, P. Van Dooren, H. W. A. M. Van Rooy, A. Aggarwal, S. Narla, and M. J. G. Van De Molengraft, "Tech United Eindhoven @Home 2022 Team Description Paper", Tech. Rep. [Online]. Available: `https://www.techunited.nl/uploads/Zorgrobots/Kwalificatie/TDP2022HERO.pdf`.

[130] M. F. van der Burgh, J. J. Lunenburg, R. P. Appeldoorn, L. L. van Beek, J. Geijsberts, L. G. Janssen, P. van Dooren, H. W. van Rooy, A. Aggarwal, S. Aleksandrov, K. Dang, A. T. Hofkamp, D. van Dinther, and M. J. van de Molengraft, "Tech United Eindhoven @Home 2019 Champions Paper", in *Lecture Notes in Computer Science*, vol. 11531 LNAI, Springer, 2019, pp. 529–539.

[131] *Vanderlande FLEET*. [Online]. Available: `https://www.vanderlande.com/evolutions/fleet/` (visited on 03/09/2023).

[132] R. Volk, J. Stengel, and F. Schultmann, "Building Information Modeling (BIM) for Existing Buildings - Literature Review and Future Needs", *Automation in Construction*, vol. 38, pp. 109–127, 2014.

[133] F. Wang and Z. Zhao, "A survey of iterative closest point algorithm", in *Proceedings - 2017 Chinese Automation Congress, CAC 2017*, vol. 2017-Janua, IEEE, 2017, pp. 4395–4399.

[134] W. Wang, B. Wang, P. Zhao, C. Chen, R. Clark, B. Yang, A. Markham, and N. Trigoni, "PointLoc: Deep Pose Regressor for LiDAR Point Cloud Localization", *IEEE Sensors Journal*, vol. 22, no. 1, pp. 959–968, 2022.

[135] D. Wilbers, C. Merfels, and C. Stachniss, "A Comparison of Particle Filter and Graph-Based Optimization for Localization with Landmarks in Automated Vehicles", in *Proceedings - 3rd IEEE International Conference on Robotic Computing, IRC 2019*, IEEE, 2019, pp. 220–225.

[136] C. Wu, T. A. Huang, M. Muffert, T. Schwarz, and J. Grater, "Precise pose graph localization with sparse point and lane features", in *IEEE International Conference on Intelligent Robots and Systems*, IEEE, 2017.

[137] G. Z. Yang, J. Bellingham, P. E. Dupont, P. Fischer, L. Floridi, R. Full, N. Jacobstein, V. Kumar, M. McNutt, R. Merrifield, B. J. Nelson, B. Scassellati, M. Taddeo, R. Taylor, M. Veloso, Z. L. Wang, and R. Wood, "The grand challenges of science robotics", *Science Robotics*, vol. 3, no. 14, 2018.

[138] S. S. A. Zaidi, M. S. Ansari, A. Aslam, N. Kanwal, M. Asghar, and B. Lee, *A survey of modern deep learning based object detection models*, 2022.

[139] J. Zhang and S. Singh, “LOAM: LiDAR odometry and mapping in realtime”, in *Proceedings - Robotics: Science and Systems*, 2014, pp. 1–9.

[140] Z. Zhou, R. Weibel, K. F. Richter, and H. Huang, “HiVG: A hierarchical indoor visibility-based graph for navigation guidance in multi-storey buildings”, *Computers, Environment and Urban Systems*, vol. 93, p. 101 751, 2022.

[141] Q. Zhu, Y. Li, Q. Xiong, S. Zlatanova, Y. Ding, Y. Zhang, and Y. Zhou, “Indoor multi-dimensional location GML and its application for ubiquitous indoor location services”, *ISPRS International Journal of Geo-Information*, vol. 5, no. 12, 2016.

List of Publications

Journal articles

- R. W. M. Hendrikx, H. Bruyninckx, J. Elfring, and M. J. G. Van De Molengraft, "Local-To-Global Hypotheses for Robust Robot Localization", *Frontiers in Robotics and AI*, 2022
- R. W. M. Hendrikx, H. Bruyninckx, H. L. Chen, J. Elfring, and M. J. G. Van De Molengraft, "From features to object patterns for indoor robot localization", *(Submitted for publication)*, 2023
- P. Pauwels, A. Van Duijven, R. De Koning, R. Hendrikx, and E. Torta, "Semantic Data from Building Digital Twins for Robot Navigation: File-Based and RealTime Data Transfer Methods", *(Submitted for publication)*, 2022
- H. Chen, R. Hendrikx, M. J. G. Van De Molengraft, and H. Bruyninckx, "Behavior adaptation for mobile robots via semantic map compositions of constraint-based controllers", *(Submitted for publication)*, 2022

Conference proceedings

- R. W. M. Hendrikx, P. Pauwels, E. Torta, H. J. Bruyninckx, and M. J. G. van de Molengraft, "Connecting Semantic Building Information Models and Robotics: An application to 2D LiDAR-based localization", in *Proceedings - IEEE International Conference on Robotics and Automation*, 2021, pp. 11 654–11 660
- R. de Koning, E. Torta, P. Pauwels, R. W. M. Hendrikx, and M. J. G. Van De Molengraft, "Queries on Semantic Building Digital Twins for Robot Navigation Citation for published version (APA): de Koning Queries on Semantic Building Digital Twins for Robot Navigation", in *9th Linked Data in Architecture and Construction Workshop*, 2021, pp. 32–42

Acknowledgments

Writing a thesis is an act of collaboration and i would like to thank a number of people for their valuable contributions, ideas and support. I would like to start with my promotors and copromotor. Herman, your expertise and enthusiasm for the field of robotics are inspiring and i have always felt welcome to discuss ideas, no matter how trivial or how complex. From our discussions i have learned not only a lot about robotics, but also about the importance of sharing ideas. René, your combination of robotics knowledge, mentoring abilities and brabantse gezelligheid have made me feel very at home in the robotics lab. Anything could be discussed during our meetings, from the newest ideas for *lazy robotics* (which, other than the name suggests, makes robots more useful), to the hurdles that go along with doctoral research, and how to overcome them. Jos, your mentorship, knowledge and pragmatic approach to combining theory and practice have been very valuable, and i have really enjoyed and learned from the discussions we have had. I would like to thank all three of you for your guidance and patience. I would also like to thank the members of the PhD committee. prof. dr. Daniele Nardi, Dr. Gijs Dubbelman, and prof. dr. Maarten Steinbuch, thank you for your time and your valuable comments.

Next, i would like to thank my colleagues from the robotics lab; Wouter, Wouter, Cesar, Ruud, Henk, Harrie, Marzieh, Yannick, Roy, Puck, Jordy, Manuel, Robert, Peter, Peter, Ruben, Danny, Koen, Busra and Elise. It was great to work with you and i have really enjoyed the past years in the lab due to the great atmosphere. A special thanks goes to Hao. We have worked a lot together over the course of our PhD journey and your kindness and enthusiasm have been awesome! I would also like to thank Elena, Pieter and Rens. While working at home during lockdown, i always looked forward to our BIM meetings and really enjoyed working on that project with you. Also, thank you Nancy and Roos for making everything run smoothly in the CST group!

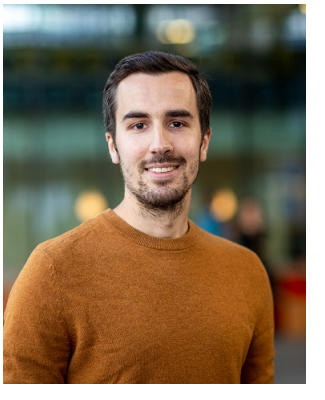
The work in this thesis was carried out as part of the FAST project, and i would also like to thank all the people involved in that project. Within the TU/e: Margot,

Hao, Liang, Gijs, Peter, Raymond, Owen, Jos and last but not least Jesse. Jesse, you started the project and have been very actively involved in making it a success. Thank you for your effort and your enthusiasm. I would also like to thank the company partners in the project: Thomas, Mauro, Roel, Don, Bas, Rene, Iwan, Jan, Elwin, Erwin, Sunniva, Burak, Tim en Kasper. The bi-weekly meetings were an opportunity to learn more about the broad field of robotics from a company perspective. And collaborating with experienced engineers from a diverse range of companies was very insightful. I would also like to thank the students that i have had the pleasure of coaching; Jad, Ioannis, Joep, Laura, Rik, Joeri, Ruben, Shubham, Bart and Spyros. Your work has provided many insights that were invaluable in writing this thesis.

Ik wil ook graag mijn vrienden bedanken voor de gezelligheid en het helpen met dingen in perspectief te plaatsen tijdens hikes, fietstochten en skitrips. Michiel, we hebben samen een groot deel van onze studies doorlopen en ik heb altijd veel aan jouw raad gehad. Joris, bedankt dat je altijd bereid bent om de hoogtepunten en uitdagingen die bij het schrijven van een dissertatie horen te bespreken. En tot slot, Papa, Mama, Anne, Huub, bedankt voor alles. Als we samen koffie drinken in de woonkamer valt alles altijd weer op zijn plaats. En Kirsten, bedankt dat je altijd voor me klaar staat, thuiskomen bij jou maakt het allemaal de moeite waard en samen gaan we door naar nieuwe avonturen.

Bob Hendrikx
Dommelen, 2023

Curriculum Vitae



Bob Hendrikx was born on April 14th, 1990 in Wageningen, the Netherlands. He studied mechanical engineering at the HAN University of Applied Science where he obtained his bachelor's degree. He obtained his master's degree in Dynamical System Design in 2017 with a focus on control systems at the Eindhoven University of Technology. As part of his studies he was a research intern at Aalborg University in Denmark, where he worked on optimal control of wave energy converters. His master's thesis was on the topic of hypertermia cancer treatment and was carried out at the Erasmus Medical Center in Rotterdam, the Netherlands.

In 2018, he started his Ph.D. research within the Control Systems Technology group at the department of Mechanical Engineering, Eindhoven University of Technology. In this project he worked on mobile robot navigation under the supervision of Herman Bruyninckx and René van de Molengraft. The research was carried out as part of the Frontiers in Autonomous Systems (FAST) project, which is a multi-disciplinary collaboration between five robotics companies and three university departments. His work focuses on robot localization and explainable robot behavior in indoor scenarios. Since December 2022 he is working for Avular as a robotics engineer.