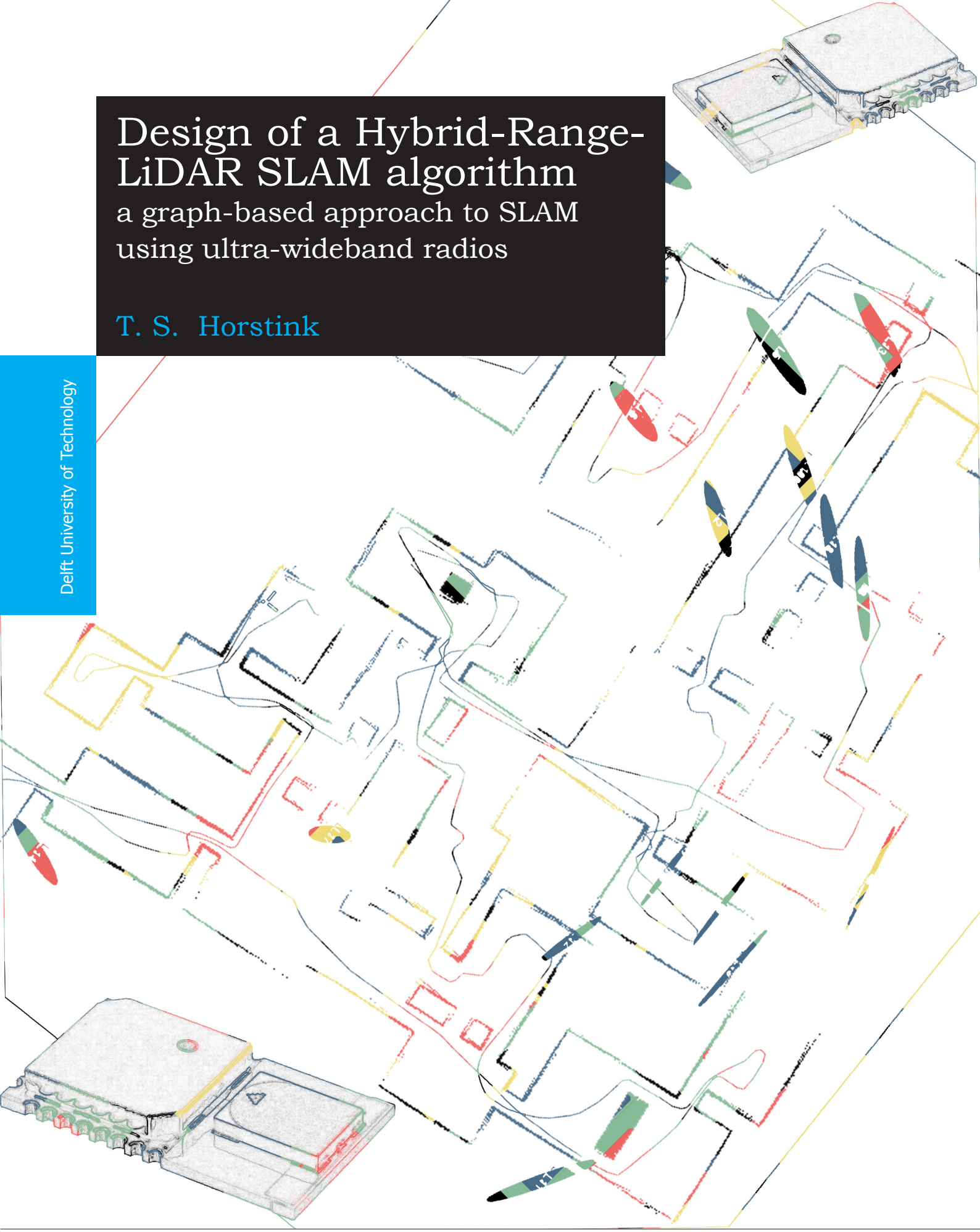
Design of a Hybrid-Range-LiDAR SLAM algorithm

a graph-based approach to SLAM using ultra-wideband radios

T. S. Horstink

Delft University of Technology



TU Delft Delft University of Technology

Challenge the future

Design of a Hybrid-Range-LiDAR SLAM algorithm

a graph-based approach to SLAM using ultra-wideband radios

by

T. S. Horstink

in partial fulfillment of the requirements for the degree of

Master of Science

in Mechanical Engineering

at the Delft University of Technology,

to be defended on Wednesday October 4th, 2017 at 14:00 AM.

Thesis committee:	Dr. ir. R. Happee,	Delft University of Technology
	Prof. Dr. ir. P. Jonker,	Delft University of Technology
	Dr. ir. J. Kober,	Delft University of Technology
	Dr. ir. A. Chandarr A.B.	Robotic Care Systems

Abstract

A fundamental prerequisite for many robot-tasks is the availability of an environment model. The ability of a robot to create such a model itself is crucial to having truly autonomous robots around in our daily lives. For basic robot-tasks, a two-dimensional grid-map that describes a distinction between inhabitable and uninhabitable space is a commonly used model. Creating grid-maps, or any environment model, is usually done by simultaneously estimating the robots location and its environment.

Modern Simultaneous Localization and Mapping (SLAM) algorithms are based on incremental non-linear optimization and the robustness highly depends on the used sensors and their proneness to environmental ambiguities [1]. In this thesis an affordable laser-range-finder (often referred to as LiDAR) is used in combination with ultra-wideband radios to improve the localization and mapping performance in large ambiguous environments.

In SLAM-literature it is common that LiDAR measurements are used to both correct the robot's ego-motion and to build a grid-map. In this thesis, however, the LiDAR is used solely for creating a grid-map while the ego-motion is corrected using Ultra-wideband (UWB) range-measurements. To quantitatively assess the mapping performance using this strategy, a Hybrid-Range-LiDAR SLAM (HRL-SLAM) algorithm was designed and implemented as a real-time application. The application is benchmarked on both synthetic- and real data and compared to similar range-based SLAM implementations.

The result of this thesis indicates that UWB radios can dramatically improve the robustness of indoor localization and grid-mapping with affordable LiDARs. Secondary results are a scalability enhancement for an existing beacon-localization algorithm, a graph-topology inspired submapping-strategy and the introduction of a reference-frame independent benchmark-metric for range-SLAM.

Finally, a few recommendations are done for improving the HRL-SLAM algorithm. The thesis concludes with an outlook on how to proceed from SLAM's graph-representation to a hierarchical metric-topological set of spatial representations, this can be seen as a tertiary result.

Acronyms

AMCL Adaptive Monte Carlo Localization.

DBN Dynamic Bayes Network.

GPS Global Positioning System.

GTSAM Georgia Tech. Smoothening and Mapping.

HMM Hidden Markov Model.

HRL-SLAM Hybrid-Range-LiDAR SLAM.

iSAM Incremental Smoothing and Mapping.

KLD Kullback-Leibler distance.

LEA Lea Elderly Assistant.

LiDAR Portmanteau of Light and Radar.

LOS Line of Sight.

MAP Maximum a Posteriori.

MCL Monte Carlo Localization.

ML Markov Localization.

NLOS Non Line of Sight.

PDF Probability Density Function.

RCS Robot Care Systems bv.

RMSE Root-Mean-Square Error.

ROS Robot Operating System.

SLAM Simultaneous Localization and Mapping.

TDoA Time Difference of Arrival.

ToF Time of Flight.

UWB Ultra-wideband.

Contents

List of Figures

1

Introduction

The world is anticipating for robots to show up in real life, especially indoors. The expectation, or at least the desire, is that robots will be able to act autonomously among humans and will alleviate us from many cumbersome tasks. Varying from daily house-chores [2] to otherwise life-threatening safety missions [3], robots will be asked to perceive and manipulate the same physical world we all live in and assist us when desired. The robot's ability to satisfactory do so depends, among other things, on how well it can address the question *how does the world look like?* Or put differently: can the robot make a sufficiently accurate model of its environment?

The robot's ability to create a model of the environment is constrained by its sensors and how it processes the data from the sensors. Where humans perceive the environment through their senses, robots perceive the environment through their sensors. Sensors are, for a multitude of reasons, prone to random variations, more commonly referred to as *noise*. The error induced by noise, especially the accumulation of it, complicates the process of building an environment model significantly [4, 5]. Aside from noise, the environment-modeling problem also grows more complicated as sensors continuously create observations that have to be processed, creating an algorithmic challenge to handle large amounts of data.

The necessary level of model complexity and accuracy, however, is determined by the task a robot is given. Common indoor robot tasks such as obstacle avoidance, path-planning and self-localization often allow a simple environment model such as a two-dimensional occupancy grid-map [6] - hence the usage of *robotic mapping* instead of *robotic environment modeling*. Figure 1.1 shows an example of an occupancy grid-map.

Robotic mapping has been covered extensively for over 30 years. Modern solutions to the problem estimate the state of the environment and the robot (its pose relative to the environment) simultaneously. The success of these Simultaneous Localization and Mapping (SLAM) systems relies heavily on a robot's capability to realize it has visited a place before [1]. This realization is used to simplify the environment model by finding its true (or at least better fitting) topology and therewith reducing the model uncertainty and complexity. This is often referred to as *loop closure*. Granted that most of the loop closures are correct, large metrically consistent, indoor as well as outdoor, environment models can be created on-line [8, 9, 10]. However, when using low-cost sensors, the ability to observe decreases significantly, introducing erroneous loop closures or missing them all together.

1.1. Contribution of the thesis

This thesis addresses the case of modeling (large) indoor environments when loop closures are difficult or unreliable and common LiDAR-based SLAM algorithms fail. This situation arises when using a low-cost 2D-LiDAR in cluttered and ambiguous environments, which indoor environments often are. In this research, we try to improve the robustness and scalability of indoor mapping and localization by

Figure 1.1: An example of an occupancy grid-map, the black cells indicate occupied cells such as walls. Grey denotes unknown space and white denotes the free spaces with no obstacles. Such maps are commonly used for navigation or localization tasks. Image adopted from [7].

introducing an extra sensor: an Ultra-wideband (UWB)-radio. Our interest lies in how this sensor can enhance the mapping and localization performance of robots in indoor environments.

The main research question can be formulated as:

How do ultra-wideband radios improve the performance in indoor robotic mapping and localization?

The question is answered through the development of a Hybrid-Range-LiDAR SLAM (HRL-SLAM) algorithm. The proposed algorithm is a probabilistic range-based SLAM algorithm that uses UWB-range-sensors in conjunction with a graph-based framework for probabilistic inference and non-linear optimization. Even though the optimization aspect is range-based, the SLAM-algorithm will still be able to create accurate grid-maps using a low-cost LiDAR.

An inherent goal to building a *new* mapping system, is the creation of a compatible localization system. It will be shown that the UWB-radio in combination with a graph representation can give rise to a localization mechanism that is natural to the graph-topology. This localization mechanism can be used as is for coarse localization or as a preprocessing step for the more common Monte Carlo Localization (MCL) method.

Secondary objectives to this thesis are maintaining a clear vision on the scalability of the algorithm and on incorporating semantics. These two objectives, albeit motivated from two different problems, computability and man-machine interaction, share the same solution as will be shown.

The novelty of this thesis is captured by:

- A range-based mapping and localization algorithm that dynamically manages spatial constraints to improve local certainty;
- A graph-topology inspired sub-mapping mechanism;
- A scalability and performance improvement to an existing spectral filtering method for beacon localization;
- A frame-independent benchmark-metric for range-based SLAM.

The algorithm is tested with data created through simulation in V-REP [11]. Verification was also done on real UWB-datasets published in [12].

1.2. Thesis outline

Chapter 2 introduces the concepts and relevant literature central to the thesis. Chapter 3 uses the provided background to sketch the design of the HRL-SLAM algorithm and elucidates design choices. Chapter 4 elaborates on the experimental set-up and chapter 5 discusses the results. Finally in chapter 6 conclusions can be found and recommendations for further research are done. The novel elements of this thesis are explained in sections 3.2.3, 3.4, 3.1.6 and 4.4. The work done towards a validation experiment using Lea Elderly Assistant (LEA) at Robot Care Systems bv (RCS) is described in appendix C. In appendix B the software structure, used parameter-values and the used libraries are presented. Appendix D shows a mapping comparison to a popular grid-mapping library.

2

Background Information

Within indoor robotics building an internal representation of the environment usually refers to building a two- or three-dimensional map. This map contains features of interest, such as the positions of landmarks or obstacles. Often the knowledge on occupied spaces and known *free* spaces are fused in an occupancy grid-map [13]. The availability of an occupancy grid-map is for many common robot tasks considered indispensable. The robot's ability to construct an accurate occupancy grid-map autonomously is therefore highly desired.

The first part of this chapter introduces a brief history on mapping and localization, focusing on the graph representation of SLAM and sequential Monte Carlo methods for localization. The middle part of this chapter deviates from graph-SLAM and introduces hierarchical SLAM and discusses how humans perceive space and how these two notions can relate. At the end of this chapter previous work in range-based SLAM is introduced and is followed by a summary of the whole chapter.

2.1. Simultaneous Localization and Mapping

Most of the research concerning the creation of maps is found in Simultaneous Localization and Mapping (SLAM)-literature. SLAM tackles the map building problem by simultaneously creating a map and retrieving the state of the robot in this map. In 2D, the robot state is a three-dimensional state of a *xy*-position and an angle for the orientation in the *xy*-plane. In 3D it is usually a *xyz*-position and the three angles describing *roll, pitch* and *yaw*. Even though it sounds more challenging to do both mapping and robot-state estimation simultaneously, opposed to *just* mapping, it is actually not. This becomes more evident if a modern formulation of the SLAM-problem is considered [14]:

> "SLAM is the process of building a map of an unknown environment while concurrently generating an estimate of the location of an autonomous robot (a moving sensor). In the most fundamental form of the SLAM problem, the only information available is from sensors on-board the robot that observe the robot ego-motion and the environment."

2.1.1. The SLAM Problem

The SLAM problem can be seen as a juncture of a motion model and an observation model. These models are probabilistically formulated in order to account for noise and the model shortcomings. The evolution of robot poses is governed by the motion model:

$$P\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t\right) \tag{2.1}$$

in which $\mathbf{x}_t$ is a stochastic function depended on the previous pose $\mathbf{x}_{t-1}$ and robot control input $\mathbf{u}_t$. Derivations and examples of such motion models can be found in [15].

Observations are used to find landmarks λ that are characterized by their constant position in the global reference frame:

$$P\left(\lambda_k \mid \mathbf{x}_{1:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}, \mathbf{n}_{1:t}\right) \tag{2.2}$$

Here λ is the set of k observable landmarks (can be a map or a set of recognizable points), $\mathbf{n}_t$ is the set of observed landmarks at time t.

In turn, the landmarks are used in conjunction with the robots state to assign a probability to the observation $\mathbf{z}_t$:

$$P\left(\mathbf{z}_t \mid \mathbf{x}_t, \lambda, \mathbf{n}_t\right) \tag{2.3}$$

It is important to note that $\mathbf{n}_t$ implies that landmarks are uniquely identifiable. While this is true using recognizable landmarks (such as identifiable radio beacons), this becomes less trivial if the landmark set is a collection of LiDAR observations. This notion is further discussed in section 2.1.5.

Combining the motion model with the observation model results in a probabilistic formulation of the SLAM problem:

$$P\left(\mathbf{x}_{1:t}, \lambda \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}, \mathbf{n}_{1:t}\right) \tag{2.4}$$

And the interest lies in finding the sets $\mathbf{x}^*$ and λ^* such that it has Maximum a Posteriori (MAP):

$$\mathbf{x}^*, \lambda^* = \underset{\mathbf{x}, \lambda}{\operatorname{argmax}} P\left(\mathbf{x}_{1:t}, \lambda \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}, \mathbf{n}_{1:t}\right) \tag{2.5}$$

From this definition it becomes trivial that the robot's own motion is an integral part of the solution to building a map. The robot needs its own motion to explore the space beyond its initial observational frame to find landmarks. Motion, however, complicates the task of properly inserting observations into a metrically consistent map. A metrically consistent map is a map that is metrically in accordance with the real world.

It is conspicuous that the joint estimation of the robot's state-evolution, in the 2D case a discrete time-series of a three-dimensional state and a map, such as a set of landmarks, gives rise to a non-linear and high-dimensional optimization problem. But an observation $\mathbf{z}_t$ is relevant to only the robot state $\mathbf{x}_t$. The projection of the observation onto a global reference frame, such as the map's coordinate frame, only requires the state $\mathbf{x}_t$. This pivotal understanding is what allowed FastSLAM [16] to reformulate the SLAM problem to a factorization of the robot trajectory $\mathbf{x}_{1:t}$ (set of robot poses) and the locations of landmarks (points in space):

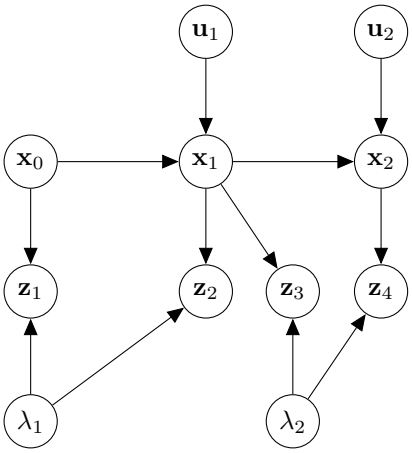
$$P\left(\mathbf{x}_{1:t}, \lambda \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}, \mathbf{n}_{1:t}\right) = P\left(\mathbf{x}_{1:t} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}, \mathbf{n}_{1:t}\right) \prod_k P\left(\lambda_k \mid \mathbf{x}_{1:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}, \mathbf{n}_{1:t}\right) \tag{2.6}$$

Here the problem is split into one estimation of the robot's trajectory $\mathbf{x}_{1:t}$ and k landmark estimation problems. This factorization is called Rao-Blackwellization and was introduced in [17]. Successful implementations such as [16, 18, 19] model the SLAM problem as Hidden Markov Model (HMM) which can be graphically represent by an Dynamic Bayes Network (DBN), an example is shown in figure 2.1a. These implementations defined the SLAM problem as an instance of Bayesian filtering and employed two filtering techniques to solve the SLAM problem: a resource-demanding particle filter for the trajectory estimation and a less demanding Kalman-filter for the landmark estimation.

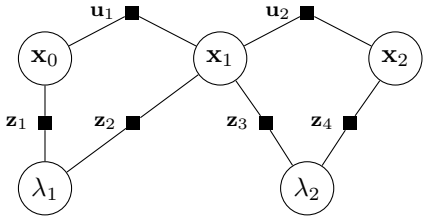
Both particle and Kalman filters share unfavorably growing computational costs once either the robot state is increased (6DOF in 3D)* or a significant growing amount of landmarks is observed (which results in a growing, dense covariance matrix†). On top of that, it was demonstrated that *filtering* itself is not the right tool for SLAM. Even in perfect noiseless conditions filtering could result in inconsistent maps [20]. These limitations induced a shift of interest towards considering SLAM as an non-linear parameter optimization problem. This concept was already introduced in [5] but was long considered to be computationally too expensive. Thanks to two independent technological advancements, the feasibility of optimizing these huge-non linear systems came within reach.

*Due to the population-based approach particle filters are generally only computationally feasible in the estimation of low-dimensional states.

†By covariance matrix, the $\mathbf{Q}$- and $\mathbf{R}$-matrices in Kalman filters are meant.



(a) SLAM representation using a Bayes Network, a directed acyclic graph.



(b) SLAM Representation using a Factor Graph, an undirected bipartite graph.

Figure 2.1: Two different representations of the same SLAM problem using two different *probabilistic graphical models*. In figure 2.1a a Bayes Network is used to: an arrow between two variables (such as from $\mathbf{x}_0$ to $\mathbf{z}_1$) indicates that the probability of $\mathbf{z}_1$ is conditional by that of $\mathbf{x}_0$: $P(\mathbf{z}_1 \mid \mathbf{x}_0)$. In figure 2.1b the factors (black squares) represent a probabilistic function of the connected variables.

2.1.2. Graph-based SLAM

Modern SLAM systems are graph-based and can be divided into two separate operational parts. The SLAM *front-end* and the SLAM *back-end*. The back-end concerns the optimization of the graph which describes the robot state evolution and the observed landmarks, this is discussed first in subsections 2.1.3 and 2.1.4. The front-end concerns feature extraction (from sensory observations) and *loop closure* which is explained in subsection 2.1.5.

2.1.3. Factor Graphs

Factor graphs are, just as DBNs, probabilistic graphical models. Opposed to DBN, a factor graph is *bipartite*, meaning there are two types of nodes: the random variables (such as $\mathbf{x}_0$ and λ_1, but *not* $\mathbf{u}_1$ and $\mathbf{z}_1$) and *factors* which connect the random variables and provide probabilistic information on these variables derived from observations (such as $\mathbf{z}_1$). In Bayes Networks, a variable *and* the accompanying PDF is depicted as one node. In factor graphs, the Probability Density Function (PDF) is separated from the variables to the special factor-nodes. What this does is that we can now have discrete variables and continuous PDFs. Figure 2.1b shows the same SLAM-problem as described with a DBN in figure 2.1a but then represented by a factor graph.

In [21] the sum-product algorithm is introduced, which is a factor graph algorithm for belief propagation. The efficiency of the algorithm allows the calculation of uncertainty propagation in the SLAM problem. Specifically the sum-product algorithm works effectively on *complicated global functions* that can be factored into *local* functions which depend on a subset of *variables*. To grasp how this applies to SLAM, the factor graph-terminology will be linked to SLAM-terminology. The theoretical grounds for modeling SLAM as a factor graph was motivated in [22].

Consider the complicated global function as a function depended on the complete robot pose trajectory and landmark positions, called the *variables*, which are constrained by *local* functions that connect poses locally to each other or to locally observable landmarks. Then, consider that these local functions are derived from observation models and the observations, the connection to factor graphs becomes more apparent.

Factor graphs, and especially the sum-product algorithm supply efficient mechanisms to perform inference on the probabilities of poses and landmarks. Also there exists a non-commutative transformation from DBNs to factor graphs, allowing the use of the sum-product algorithm unconditionally. In

the end, both representations are probabilistic graphical models that describe the same factorization as in equation 2.6.

2.1.4. Graph Optimization

The second challenge is finding the optimal values for the *variables* as the state-vector $[\mathbf{x}, \lambda]^\intercal$ has now grown significantly in this definition, containing all the robot-poses and landmark-positions. The objective function that is to be optimized is based on the maximum likelihood of the earlier mentioned *complicated global function*. Sensor observations usually serve as the initial estimates for the optimization. Now it happens to be that optimizing the factor graph of a DBN is *exactly* the same thing as finding the MAP for that same DBN, and it can be done using the max-product algorithm described in [21].

There is a vast body of literature concerning the optimization of sparsely connected systems. If the *system*-matrix $\mathbf{A}$ as in $\mathbf{A}x = \mathbf{b}$ describes a sparsely connected system, then $\mathbf{A}$ is a sparse matrix*. Based upon [22] hyper-graph optimization libraries such as [23, 24, 25] are able to solve SLAM problems of thousands robot poses and constraints† in just mere seconds using Gauss-Newton, Levenberg–Marquardt or Powell's Dogleg method. These hyper-graph optimization libraries exploit the sparse structure of the SLAM-problem and use QR-decomposition or Cholesky decomposition to efficiently invert huge system-matrices‡. For a detailed explanation on non-linear optimization of SLAM, the reader is referred to appendix A.

Remarks on Graph Optimization and SLAM

The optimization approach increases the consistency of SLAM [15] and this is mostly due to the continuous recalculations of the Jacobians around the current set of optimal variables [22]. Implementations such as [8] increase the efficiency of the optimization by only re-linearizing the Jacobians once a certain threshold is reached. Also in [26] an incremental version of Powell's Dogleg method is introduced that attains a speed boost with negligible loss of accuracy compared to Levenberg–Marquardt.

Despite the efficient approaches for non-linear optimization, there lies a general pitfall. Non-linear optimization is prone to convergence to a local minimum. Especially once the state covers a thousand dimensions (say, 300 2D robot poses and 50 2D landmarks) it becomes hard to belief that the found solution is also the actual set of robot-poses and landmark-positions that results in the MAP. Literature, however, reveals that non-linear optimization of SLAM often does lead to a global optimum.

The success of non-linear optimization converging to a global optimum relies heavily on the quality of the initial estimates. Luckily, in SLAM locally accurate initial estimates are available through observations as, odometry, LiDAR, vision, etc. generally create locally accurate estimates. The incremental nature of the optimization problem allows to add accurate pose increments to an optimized recent pose, highly improving the chance of successful convergence to a global minimum. This is discussed and exploited for linear approximations in [27, 28, 29] and is proposed in [14] as a lead to why SLAM so often successfully converges to a global optimum.

2.1.5. Loop closure

An aspect that improves the quality of robotic mapping significantly is the robot's ability to *realize it has visited a place before*. In literature this is often referred to as *loop closure*. Loop closure owes its name from the case where a robot drives around in a loop and eventually recognizes it as the beginning of the loop where it initially left off: *closing the loop*.

If a robot is unable to recognize it is at a place it has seen before, it can not close the loop while building a map. As a consequence, the map would grow infinitely large as the robot continues to

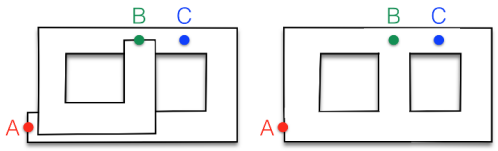
*A sparse matrix is a matrix in which most of the elements are zero.

†It should be noted that *constraints* in the case of SLAM are also probabilistic. The chance of finding the set of variables that complies with all constraints is negligible small. Therefore, the interest usually lies in minimizing the accumulated violation of all the constraints.

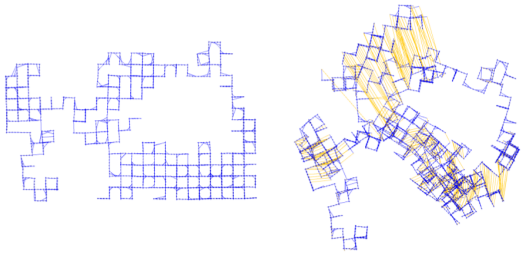
‡The actual *trick* of these decompositions techniques is that the true inverse is not calculated at all.

wander around. Each observation is considered as an genuinely new piece of the environment and stitched to the current *location* of the robot, creating an growing chain of observations. From a graph-perspective: the world becomes an infinitely long hallway, as depicted in figure 2.2a. It is obvious to see that for navigation purposes alone, the absence of loop closures degrades performances significantly as possible *shortcuts* will not be found.

Secondly, *correct* loop closures bring a great deal of robustness to the graph optimization discussed in section 2.1.4. As a robot explores, the uncertainty of a robot's trajectory, measured through odometry will grow without bounds. Eventually, the odometry derived trajectory will look nothing like the ground-truth trajectory. The path can be corrected by incorporating constraints that can imply $x_1 = x_{10}$ for instance. This basically closes the loop. Figure 2.2b demonstrates how corrupted odometry input can be corrected by *correctly* incorporating loop closure constraints in the optimization process. The residual created by a loop-closure constraint will generally be significantly larger compared to odometry-induced residuals. Loop-closure constraints will therefore affect the optimization gradient significantly more allowing recovery.



(a) Correct loop closures can reveal the true topology of the environment. Figure adopted from [1].



(b) Correction of pose-graph using loop closure constraints. On the right the raw odometry path is shown with yellow loop closure constraints. On the left the optimized trajectory is shown. Figure adopted from: [30]

Figure 2.2: Implications of *loop closure*, the ability to recognize known places are revisited.

Loop closure is a difficult problem as observed features are not necessarily uniquely identifiable. Loop closure research falls into the category of *data-association*, which concerns associating a current observation with past observations. Data association is considered an NP-hard* problem [4].

A 2D-LiDAR scan in a square room can easily be mixed up with an other scna in a different square room. This can be overcome partially by introducing concepts such as *map-matching* as implemented in [31]. In [31] maps are matched instead of single scans. A map, a collection of scans, contains more features than a single scan reducing the number of ambiguous situations. Still, this is only a palliative as such a system would still fail in metrically equivalent environments such as two very similar floors in a flat. Secondly, it also introduces the design-question: how big can local maps be? When do maps become metrically inaccurate? These questions have no straightforward answers and are often left as a configurable parameter.

Other approaches use SIFT† -features to generate loop closure hypotheses. A robot's vision is used as input for the SIFT algorithm to find features and then evaluates whether it has seen this feature before. But in this case, SIFT-features are still difficult to uniquely identify and we do not even consider the computational burden of processing video or photos.

Another main problem of loop closure lies in the computation: the computational feasibility of matching previous observations to new observations grows as the amount of features grows (due to discovery of new features or simply because a larger environment is discovered). This can be partially

*NP-hardness is a measure of computational complexity and stands for *n*on-deterministic *p*olynomial-time hard, basically implicating there are no efficient ways to compute an answer to the problem.
†Scale-invariant feature transform (SIFT) is an algorithm used in computer vision to detect and describe features in images.

overcome by only matching to features that fall within a pose's certainty region. However, if the map grows, or more difficult to distinguish features arise, the chance to introducing *false* loop closure constraints increases significantly none the less.

2.2. Localization of Mobile Robots

In SLAM, the robot's current pose, in a local frame, can be accurate. But the uncertainty of the robot's pose increases once it moves. The increase of uncertainty can lead to the situation in which a robot loses the knowledge of its location relative to its reference frame*. For navigational tasks one must often know a start- (the robot's current pose) and an end-location (the navigational goal) in the same reference frame†. If the current robot pose is unknown (or too uncertain) navigation becomes impossible. Therefore, the ability to recovery from lost localization is crucial to robot autonomy.

This section will consider recovery of localization if no prior information is known but a map is available. This is also known as the *kidnapped robot problem* in which an (autonomous) robot in operation is moved to an arbitrary location.

Robot localization is in general described as an instance of the Baysian Filtering problem [15]. Which can be formulated as finding the most likely state, the MAP, $\mathbf{x}_t^*$ given observations $\mathbf{z}_{1:t}$:

$$\mathbf{x}_t^* = \underset{\mathbf{x}_t}{\operatorname{argmax}} P\left(\mathbf{x}_t \mid \mathbf{z}_{1:t}\right) \tag{2.7}$$

2.2.1. The Localization Problem

When the robot is to be localized, one is usually interested in the state of the robot at a time t. If $t > 0$ there is an initial state $\mathbf{x}_0$ and a set of observations $\mathbf{z}_{1:t}$ available. Recursively calculating from t_0 up to t using equation 2.7, the MAP $\mathbf{x}_t^*$ can be found. One step in the recursive estimation scheme consists out of two phases. The *prediction*-phase and the *update*-phase.

The prediction phase uses a motion model, such as equation 2.1 to predict $\mathbf{x}_t$ given only the previous state $\mathbf{x}_{t-1}$, the previous control input $\mathbf{u}_{t-1}$ and observations $\mathbf{z}_{1:t-1}$, predicting $P\left(\mathbf{x}_t \mid \mathbf{z}_{1:t-1}\right)$. The probability distribution function can be expressed as the integral over the possible previous states‡:

$$P\left(\mathbf{x}_t \mid \mathbf{z}_{1:t-1}\right) = \int P\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_{t-1}\right) P\left(\mathbf{x}_{t-1} \mid \mathbf{z}_{1:t-1}\right) \mathrm{d}\mathbf{x}_{t-1} \tag{2.8}$$

The update phase uses an observation model, $P\left(\mathbf{x}_k \mid \mathbf{z}_t\right)$, to integrate sensory information from the sensors to build a probability density function of the current state $\mathbf{x}_t$:

$$P\left(\mathbf{x}_t \mid \mathbf{z}_{1:t}\right) \tag{2.9}$$

$\mathbf{z}_t$ is assumed to be conditionally independent of $\mathbf{z}_{t-1}$ given that $\mathbf{x}_t$ is available, this is the same assumption that considers the projection of independent observations into a global reference frame as described in section 2.1. Analog to equation 2.3 the probability of an observation can be assessed through $P\left(\mathbf{z}_k \mid \mathbf{x}_k\right)$. Finally, applying Bayes' theorem, the PDF can then be described by:

$$P\left(\mathbf{x}_t \mid \mathbf{z}_{1:t}\right) = \frac{P\left(\mathbf{z}_k \mid \mathbf{x}_t\right) P\left(\mathbf{x}_k \mid \mathbf{z}_{1:t-1}\right)}{P\left(\mathbf{z}_k \mid \mathbf{z}_{1:t-1}\right)} \tag{2.10}$$

The algorithm can be sequentially repeated in the order *predict* and then *update* up until the state $\mathbf{x}_t$ is reached - and the robot is *localized* once again.

*In robotics, the position in the map-frame is usually referred to as the *global* position.

†Navigation in robotics often refers to applying some kind of A^* or Dijkstra's algorithm to find the shortest path between the start and end on either an occupancy grid-map or a topological map.

‡In this definition it assumed that $P\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_{t-1}\right)$ - which implicates that the state evolution $\mathbf{x}_{0:t-1}$ is fully encoded in $\mathbf{x}_{t-1}$, this is called the *Markov property*. The *Bayes filter* depends on the Markov assumption. In [15] the justification of this assumption is discussed and considered often a proper approximation.

2.2.2. Monte Carlo Localization

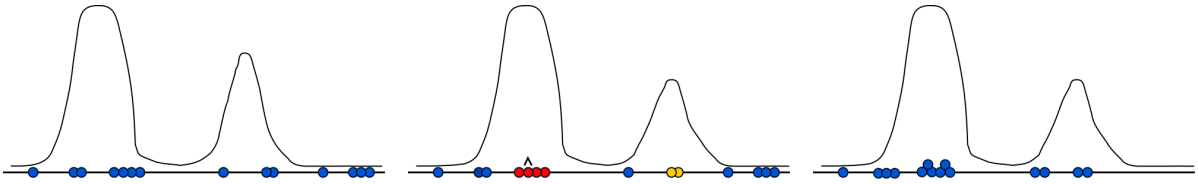
MCL was introduced in [32] and uses a sequential Monte Carlo method (e.g. particle filtering) to evaluate the probability distributions. The algorithm was improved noteworthy in [33]. Initial attempts to solve the localization problem were approaches based Kalman-filters and *Markov Localization*[*]. MCL introduces advantages over Kalman filtering and Markov Localization:

- MCL can cope with multi-modal probability distributions, opposed Kalman-filters which are restrained to Gaussians;
- Small computational footprint compared to ML;
- High frequency of observation integration.

In MCL a *particle population*, S_t, is used to represent the probability density $P(\mathbf{x}_t \mid \mathbf{z}_{1:t})$. The population consists out of N particles and each particle $s_t^n \in S_t; n = 1, \ldots, N$ represents a *hypothesis* for $\mathbf{x}_t$. If a weight proportional to the *quality* of a particle s_t^n can be assigned, S_t can be used to sample efficiently from a discretized representation of the PDF $P(\mathbf{x}_t \mid \mathbf{z}_{1:t})$, this is called *Sample/Importance Resampling* (SIR) and was proposed in [34] to obtain samples from difficult to sample PDFs.

To recursively estimate $P(\mathbf{x}_t \mid \mathbf{z}_{1:t})$, at each time-step t, samples are to be drawn from $P(\mathbf{x}_t \mid \mathbf{z}_{1:t})$ to process the predict- and update-steps. These two steps are followed by resampling which ensures preservation of the descriptive relationship between S_t and $P(\mathbf{x}_t \mid \mathbf{z}_{1:t})$. Resampling *keeps* the most likely hypotheses in S_t while the worst hypotheses are discarded from S_t. In favor of the discarded particles, new (random or quasi-random) particles are added to S_t. The new particles are inserted to preserve a diverse population that can maintain a certain robustness to observational and modeling errors, preventing a rushed convergence to an unimodal distribution. Resampling happens after all particles are assigned a *weight* w_n describing its fitness using the update model. The particle with the highest weight is considered the MAP: $\hat{\mathbf{x}}_t^*$

The combination of the discretized representation of the PDF and resampling techniques is commonly referred to as particle filtering. The reader is referred to [32] for the theoretical justification that $S_t \approx P(\mathbf{x}_t \mid \mathbf{z}_{1:t})$. Figure 2.3 describes one iteration of the recursive estimation visually.



(a) S_{t-1}. S_0 is often initialized as an uniform distribution.

(b) S_{t-1} updated to S_t according to 2.8 and weighted using 2.10.

(c) S_t after re-sampling.

Figure 2.3: Visualization of one iteration in the recursive estimation of $P(\mathbf{x}_t \mid \mathbf{z}_{1:t})$. The solid black line represents the *true* but *unknown* probability distribution. The dots represent the particles used to represent the distribution. The color of the particles describe the relative alikeness: red is most probable, than yellow followed by blue. The most likely particle is denoted by ^. Image modified from [35].

2.2.3. Localization through Factor Graphs

A different approach is to localize the robots global location using factor graphs and non-linear optimization. This approach, however, is only applicable if the factor graph contains landmarks that *anchor* the graph to a *world* and are *easily* observable without a huge computational burden. A set of relatively constrained poses (such as in a pose-graph without landmarks) and no prior information on an initial

[*]For a short introduction to Kalman- and ML the reader is referred to [32].

estimate results in a tremendous grid search through a (possibly) complex search space: a recipe to get stuck in a local minimum.

Using landmarks, the search space can be made smaller by only searching space near the observed landmark. An example is given in [24] where GPS-factors, which encode an absolute position, are used to constrain poses, *localizing* the robot.

Note that non-linear optimization in *just* localization works significantly different from SLAM as there is *no* decent initial estimate nor a comparable problem-structure. The success of non-linear optimization as localization mechanism depends highly on preprocessing of observations to create an accurate initial guess and non-complex search spaces. These two challenges share many challenges with data-association. Localization through non-linear optimization is therefore in the case of a LiDAR equipped, kidnapped robot, unfavorable compared to MCL.

2.3. Discussion on Robustness and Scalability

Up until now, modern implementations of both a SLAM- and a localization-algorithm have been introduced. Both are probabilistic algorithms and are successful because the math exploits the structure of the problem well. In the case of SLAM, the non-linear optimization and factor graphs can often create a consistent map. Secondly, the computation of belief propagation is highly efficient using the sum-product algorithm and can be used to shrink the search space for loop closures. In the case of global localization, the estimation of a low-dimensional state from a complex multi-modal probability distribution is straightforward and on-line computable by virtue of MCL.

The fact however, that autonomous robots are not yet moving around through our daily lives, is a confirmation that these algorithms are not mature enough for complete autonomous robot behavior. This section discusses open standing issues in SLAM and global localization.

2.3.1. Challenges in Mapping

While the back-end of SLAM algorithms, the graph-optimization and belief-propagation parts, describe and compute the SLAM problem very efficiently, effectively and accurate. It is the front-end that lacks robustness and introduces errors that even a well designed optimizer cannot solve.

Robust optimization and outliers

The front-end is responsible for creating relations between poses and landmarks. It is evident that if the front-end makes mistakes, *outliers*, such as recognizing of one object as another or the recognition of two different instances of the same object are introduced. Outliers become apparent in the form of erroneous constraints do not necessarily describe the real world. The common approach in graph-SLAM, which minimizes a least squares cost-function to maximize the likelihood, *assumes* that the residuals of all constraints have a Gaussian distribution. It is evident that wrong loop-closures and sensor-noises rarely have a truly Gaussian distribution, rendering the assumption faulty. On top of that, least-square estimators are known to be very prone to outliers, and can therefore corrupt the result of the optimization process significantly.

Modern SLAM systems try to tackle this problem by making the topology *part of the optimization*. This is done by using more tropical M-estimator functions that try to compensate for the erroneous Gaussian assumption. In [36] constraints can be switched on or off by introducing a new factor. In [37] constraints are not considered right or wrong but their error is characterized in a more expressive way, resulting in a Bayesian framework solely for the constraints. [38] is also based on switchable constraints but does it such that an extra factor is unnecessary. [39] approaches the problem by only adding constraints to the optimization problem if the constraint is consistent with other constraints, this is achieved using spectral partitioning. These techniques are reviewed and compared in [36] but no satisfiable conclusion is drawn: using synthetic data yields reasonable performance, real-world data experiments yield significantly less success. In [40] the usage of M-estimators is encouraged but it also acknowledges that it takes knowledge on the behavior of the cost-function to properly pick and tune

a M-estimator. Figure 2.5 shows how the combination of bad initial estimates and wrong constraints can introduce a difficult starting point for the optimization. Figure 2.4 shows four typical estimator functions.

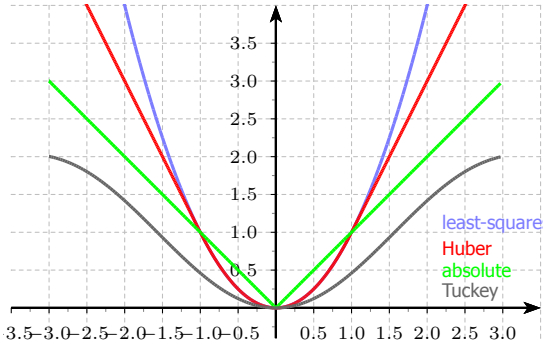


Figure 2.4: Cost function of four different M-estimators. Huber's and Tuckey's (also known as bisquare) cost functions are dependent on a parameter k which needs to be picked. In this plot $k_{huber} = 1$ and $k_{tuckey} = 3.5$ are chosen such that it visualizes the characteristic differences between the cost functions clearly.

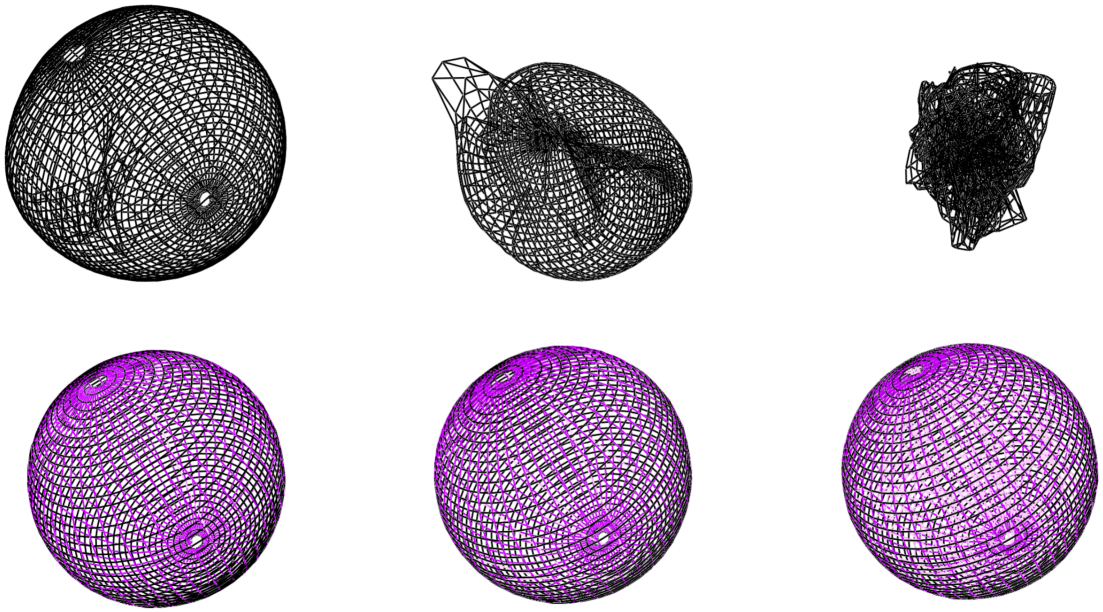


Figure 2.5: The top row shows the initial state of the distorted Sphere2500 dataset. The bottom row shows the optimization using [37] recovering the original topology. Each column has a different amount of outliers, the most right sphere has the most outliers. Successful topology recoveries such as these are not uncommon, yet not reliably enough to consider it *robust*. Image from [37].

Scalability of SLAM

Scalability is an open problem in SLAM [1]. Since the environment is big, in essence it could be the whole world, or even universe, it is a valid concern that collected data, either processed or unprocessed, will exceed a storage limit at some point and render the model unusable. Even if the environment is constrained, e.g. the robot stays within a building, it will still accumulate an infinite amount of data over time as long as the robots observes.

Various approaches improve the scalability by removing variables from the factor graph based. This

is done wile trying to represent the maximum amount of information (of the original graph) using minimal number of variables [41, 42, 43]. Effectively this means that observations from the robot's belief are *eliminated*. This decreases the amount of variables but also reduces the sparsity of adjacency-matrix that represents the graph [41].

Assessing the quality and topology of a map

Maybe one of the most fundamental problems in SLAM is that the quality of the result is not easily measurable. Especially, as often the case with SLAM, no ground-truth is available [44]. Error functions do not always describe the error relative to the real world: it can also be the error relative to some distorted map. In [4] is it showed how different error functions, such as the Mahalanobis distance*, do not give consistent conclusions on the quality of the solution. In [44] an approach is suggested to define errors locally and assess the performance on the mean of all local errors compared to a ground-truth.

This still does not solve the problem fully. How can can the quality of the found topology be assessed? In figure 2.5 we *know* the result should be a sphere but if a robot is exploring an unknown space, and there is no ground truth, there is no reference to assess the quality. The problem is further complicated as SLAM is fundamentally *unobservable*†. The observability of the SLAM problem is partially gained by introducing assumptions on prior information or by introducing a fixed reference frame, this still leaves the topology not assessable.

2.3.2. Challenges in Localization

The *front-end* development of MCL has progressed and many different probabilistic models for the update-step have been proposed based on a variety of sensors: bearing-only sensors, range-only sensors, LiDAR and even fingerprint based approaches using radio-receivers. The main challenges in MCL are not necessarily about about influence of the quality of perception, but more on the drawbacks of maintaining a discretized approximation of a continuous PDF. The quality of the approximation can degrade if the resampling step removes unlikely particles prematurely. It is not guaranteed that the most likely particle at each iteration is also the MAP; that is why MCL is an iterative algorithm. Therefore, the chance exists that the MAP particle is removed from the discretized distribution *before* it becomes apparent that it actually is the MAP. This can let MCL converge to an incorrect hypothesis. There are mechanisms to lower the chance of this premature convergence occurring, but there is no guarantee. These mechanisms include the addition of random particles, re-initialization of the MCL or, most popular, the use of an adaptive population. In [46] the concept of an adaptive population is proposed and it grows and shrinks the populated based on the Kullback-Leibler distance (KLD), this is considered a good error-function in [46] and has had numerous good result. The popular Adaptive Monte Carlo Localization (AMCL) Robot Operating System (ROS)-package is based upon [32, 33, 46] and provides reasonably robust performance but still no guarantees.

The second challenge in MCL is the population size. MCL works for low-dimensional state-estimation because the search-space is relatively small. A 3D search space (position and orientation in 2D) of a moderately sized environment (say, 25 squared meters), can require about a 1.000 particles to reach a certain convergence. The population size grows exponentially with the state-size if the same particle density is desired. This can already become problematic for a full state recovery with no decent prior estimate in 3D (which has a six dimensional search space). This limitation bounds the scalability of MCL considerably.

These two limitations, robustness of the approximation and scalability of the population, limit the use of MCL to *small* maps *or* maps with specialized initialization routines, using prior information to create a specific initial population.

*"The Mahalanobis distance is a measure of the distance between a point P and a distribution D" [45].

†The concept observability originates from control theory and describes the ability to compute a unique state from a series of control input (e.g. $\mathbf{u}_{1:t}$) and observations (e.g. $\mathbf{z}_{1:t}$).

2.4. Human Perception of Space and Human-Robot Interaction

Human performance in mapping, localization, navigation and obstacle avoidance is in general superior to that of robots. Exceptions, however, occur: car navigation systems can *outperform* humans because there is almost always a reference to a global frame due to availability of GPS: it is hard to get lost that way. In the absence of a global reference frame, robots seldom match the performance of humans when it comes to localization and mapping. The lack of a *unique*[*] global reference frame is a *fundamental* property of mapping and localization problems, yet humans seem to cope without problems. Can analysis on human perception improve robotic mapping and localization?

Biology often serves as a source of inspiration for technology. Biology, more specifically, humans, can serve as a source of inspiration for improving robots to cope with mapping and localization when there is no global reference frame. There is also an additional reason why to take interest in human perception and modeling of space: it is relevant to how robots and humans interact.

2.4.1. Human mapping and localization

There is at least one thing that humans and mobile robots have in common when it comes to mapping and localization: the estimated traversed distance error becomes more erroneous as the traversed distance increases. With 2D-robot poses this error is approximated to grow at most linearly [47] and for humans it can be a lot worse. As been mentioned earlier, the accumulation of all these ego-motion errors complicates mapping and localization.

Luckily, humans take much less interest in precise information such as traversed distance or global orientation when it comes to building spatial representations opposed to common mapping algorithms. Humans use a combination various spatial models to map spaces, localize and navigate in them [48]. Humans also partition spaces using *semantics*[†]. This is efficient because a lot of indoor environments share similar features and anatomy: rooms are connected through hallways and doors common objects include beds, toilets etc. Specific rooms can be identified by the objects they contain: if there's a bed, it is usually a bedroom. Humans do not use occupancy grid-maps to assign an identifier to a room, they use semantics and hierarchical concepts to differentiate between different spaces.

In localization the semantics are also used effectively. A *global estimate* can be deduced from knowledge as in *I'm at work, I work at company X, therefore I am in a certain building* and *I work on the top floor, I observe my desk, therefore I am on the top floor*. This estimate can be locally refined by observations: *I see my computer screen, I am therefore sitting at my desk* of which I know it is at a fixed position. But also if the third floor has never been explored, a human can know it is there (how often is there a fifth floor and no fourth floor?), and once explored, it can be attached to the existing model of the work-building.

The question arises, how do humans store perceptions and models of space? Research on this subject has mostly been conducted from a perspective on how quickly humans can learn new environments and how tools such as a map improve the learning curve. Some of the research has tried to explain the results by arguing how humans make internal representations of space. In [49] two hypothesis are considered: Thorndyke's hypothesis which argues that humans do keep the euclidean structure of space preserved in their models [50]. The opposing hypothesis from Stevens en Coupe which suggests that humans do not preserve the euclidean structure of space when they make a spatial model [51]. The interesting aspect about [49] is that it brings up a framework of *memories* that contain multiple layers encoding different kinds of information: some layers encode euclidean properties and some do not. Suggesting that both [50] and [51] are right in their own way and McNamara coins this as a form of human hierarchical mapping and it is remarkably how well this rhymes with some attempts in robotic mapping that incorporate semantics [52, 53].

More recent research on human space-perception goes into more specific aspects of properties such as what kind of abstractions are done. In [54] it is suggested that spatial memories of small

[*]The term unique is used because the reference frame is *often* defined as the first pose, yet, nothing prevents from defining an other pose as reference frame: there is no fixed reference frame.

[†]Semantics, philosophically, is the study of meaning. It focuses on the relationship between significant objects — such as, houses, rooms, furniture

maps improved if there was information on relative orientation while memories of large maps did not benefit of information on orientation. This is an interesting point as it can be interpreted that more information does not always gives a performance boost - an idea that is used to promote the topological maps discussed in section 2.5.

2.4.2. Towards mutual spatial understanding

A specific subset of mobile robots is the set of robots that is expected to interact with humans. For these robots, task formulations using human semantics is *the* elusive pipe-dream. In the case of navigational tasks, *go to the kitchen* is preferred over supplying x and y coordinates. This can only be achieved if there exists a translation between the robots internal representation of space and the spatial models humans use.

It is obvious that there is no perfect translation between these internal spatial representations. Even between two humans errors are not uncommon. The annotation of a partitioned map, based on available features that are semantically described opens up the door to *basic* man-machine interaction. Allowing parts of a robot-map to be annotated with semantic features creates a start for at least the robot understanding human spatial representations. Labeling features semantically is not a new research area. In semantic mapping there are [52, 55, 56] in which rooms are classified using extracted features and vector-support-machines. These feature-clusters are topologically connected if a robot can traverse between them - creating a topological map that holds semantics. Also more general, companies such as Google, or at a smaller scale, Blippar can already reasonably assign human terminology to features on-line using (monocular) cameras. Figure 2.6 illustrates an example. If such annotations are added as meta-data to poses there is a new range of techniques that can be used to bring new, not necessarily metrically, structures to maps.

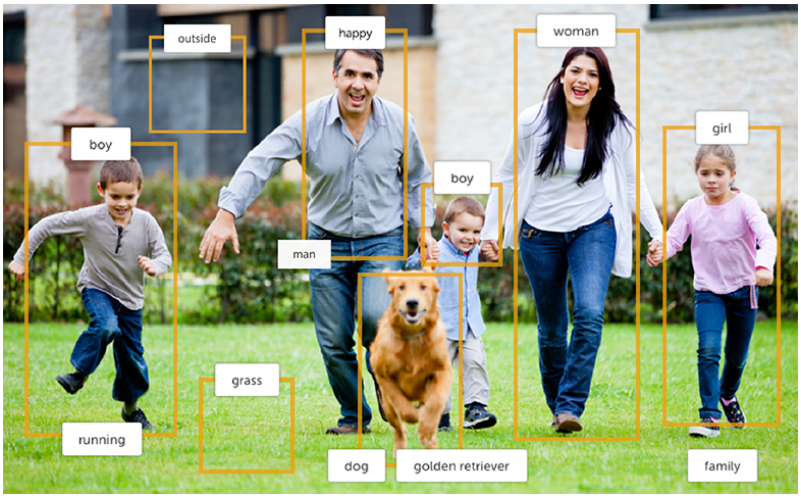


Figure 2.6: An example of Blippar's API assigning labels to input images. Image adopted from Blippar's website [57].

2.5. Hierarchical Mapping and Localization

Hierarchical-, *topological-*, *sub*-mapping and many other similar terms have been coined in SLAM research to describe methods that split the SLAM problem in to a set of smaller tractable problems. In almost all definitions there exists at least two models describing spatial relationships differently. The motivations to split the SLAM problem, or partitioning the environment model, was historically motivated to enhance the algorithmic scalability. Nowadays, especially for indoor robotics, the scalability of graph-SLAM itself is sufficient for many situations. However, in the case of man-machine interaction and the desire for improved mutual spatial understanding, *topological*-maps may have a new goal.

Hierarchical Mapping in SLAM

There are two commonly seen approaches for hierarchical mapping. Filter-based SLAM used hierarchy to create a map that governs smaller maps (sub-maps). Breaking the SLAM problem into smaller problems was a necessity for filter-based SLAM to be ran on-line. The justification is based on the assumption that the SLAM problem is easily solvable for small environments. If these small environments can be captured by sub-maps, the new problem becomes finding a topology that can govern the sub-maps. The new topology would not have to capture the full metric detail but only a global structure that exists between the sub-maps. Through the global structure only a set of sub-maps would have to be loaded, improving the scalability and accuracy of filter-based SLAM. Excellent examples are Bose's ATLAS framework which was presented in 2004 [58] and Blanco's *Hybrid-Metric-Topological* (HMT) framework in in 2008 [59]. ATLAS created sub-maps based on pose-uncertainty and Blanco applied spectral partitioning to find maps that contain sets of poses with consistent and coherent observations: if a set of poses can not describe the same environment, they are probably not in the same map. The size is these maps is also picked arbitrarily or set as a configurable parameter.

Graph-SLAM removed the necessity for hierarchy in a sense that graph-SLAM can be run on-line for huge graphs. Yet, *hierarchy* was re-introduced, as a term for a slightly different concept. The motivation was again with a focus on computational relieve in order to improve the scalability of graph-SLAM. Examples include creating a sparse graph, based on the original graph such as done in [9] and depicted in figure 2.7 or by introducing new graphs that are based on minimally interdependent sets of variables: the Thin-Junction-Tree-Filter [60] and its spiritual successor the Bayes tree [61]. The difference between [9] and [60, 61] is that the prior removes variables to simplify the optimization and covariance recovery, the later approaches focus on keeping the variables but by re-ordering them in such a way that the computation only involves the affected variables*. Google's Cartographer SLAM-algorithm [31], uses a map-graph, in which small maps are constrained by rigid transformations derived from odometry and loop-closures. This is much like the filtering-based hierarchical mapping approaches but combined with a graph-approach.

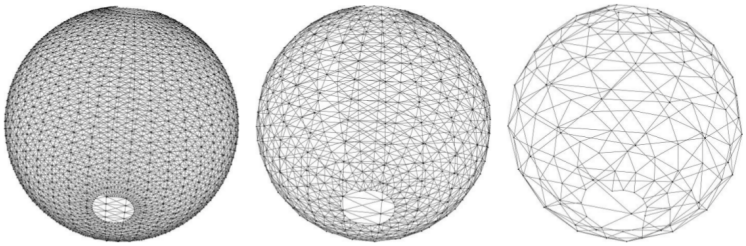


Figure 2.7: An example of three *different*, spatial representations of the Sphere2500 dataset. *Different* in this case means *simpler*. Image adopted from [9].

2.5.1. Hierarchy: a Basis for Semantic Mapping and Localization

For this thesis, the term *hierarchical map* is used to describe a model that holds an arbitrary amount of spatial representations that are somehow linked. For example, a graph in which each vertex represents a room and an edge represents the property you can traverse between the two rooms, then the whole graph can be seen as a spatial representation of a collection of rooms, which could be coined a building. The fundamental property of hierarchical maps is that there are spatial models at different levels of complexity or abstraction. This property is desired as such information is essential to do *cheap* inference for localization: if you know you are in a specific room, you do not need to search the whole building. This is also where a parallel can be sought to humans: humans do inference on their whereabouts and use concepts such as countries, cities and streets to shrink the search space by adjusting the *resolution* and not its *size*. Hierarchical maps that have various search-spaces at different levels of complexity enable us to change a large-scale localization problem from a needle-in-a-haystack situation to a feasible inference problem.

*This process, using the Bayes tree, is neatly visualized in https://www.youtube.com/watch?v=R47oeNAatLI.

2.6. Description of Ultra-wideband Sensors

An UWB sensor usually refers to a radio-based range-sensor. Two UWB-transceivers are needed to calculate the distance between them, the process is called *ranging*. Other commonly used range-sensors are based on optics (LiDAR) or acoustic (hypersonic range-sensors) and measure the range to any object instead of another sensor. All three sensors types, shown in figure 2.8, have different form-factors, work at different frequencies, have different drawbacks and deliver different accuracy. UWB range-sensors can measure ranges up to 250 m, outdoor with clean a Line of Sight (LOS), by being able to transmit information between two sensors by sending messages. UWB transceivers are relatively new. It was only in 2005 that the Federal Communications Commissions (FCC) designed standards for UWB communication systems. The regulated UWB standard sparked the development of UWB systems, resulting in the availability of UWB-radios at low prices since 2012.

(a) DWM1000 (± €15) **(b)** RPLiDAR (± €200) **(c)** Hypersonics (± €2)

Figure 2.8: Three different range sensors, from left to right based on: radio (UWB), optics (laser) and acoustic (hypersonic).

The interest in UWB technology has grown in the past years as it has proven to be a very usable tool in localization and SLAM tasks [62, 63, 64]. UWB's most notable feature is that it uses sub-nanosecond duration of pulses of several GHz bandwidth. This offers the ability to distinguish different multi-path components of the signal. Also these pulses do not interfere with other radio-systems that occupy the same frequencies (such as WiFi or Bluetooth). The ability to distinguish multi-path components is a major advantage when used for Time Difference of Arrival (TDoA) distance calculations.

For this research, UWB-sensors, with a fixed antenna and Decawave DW1000 chip, developed in-house by RCS are used. The boards have demonstrated to be able to range up to at least 25 m with an accuracy of 1% indoor. The observations can be done with a frequency of about 5 Hz. The ranging method is based on the *two way ranging* protocol from Decawave which is optimized for distance measurement calculation based on the IEEE 802.15.4a standard using two asynchronous clocks. Long-term clock synchronization while requiring picoseconds accuracy is infeasible, therefore multiple messages are used derive the Time of Flight (ToF) data from two asynchronous clocks.

The protocol is graphically depicted in figure 2.9. It is based on the exchange of three messages between two UWB-transceivers, usually called beacons, and more specifically as a tag and an anchor. The protocol has three different types of messages: the *poll*-, the *response*- and the *final*-message. The calculation is based on the timestamps T_{SP}, T_{RR}, T_{SF} gathered by the tag and the timestamps $T_{RP}, T_{,SR}, T_{RF}$ gathered by the anchor. These timestamps indicate the moments of sending or receiving one of the aforementioned messages. The ToF can be approximated by equation

$$ToF = \frac{1}{4}\Big((T_{RR} - T_{SP}) - (T_{SR} - T_{RP}) + (T_{RF} - T_{SR}) - (T_{SR} - T_{RR})\Big) \tag{2.11}$$

followed by the distance calculation $d = ToF \times c$, where the speed of light (in air 299 700 000 m s^{-1}) times the ToF gives the distance d between the tag and the anchor. An additional *report*-type message can be used to transport the distance information to the tag.

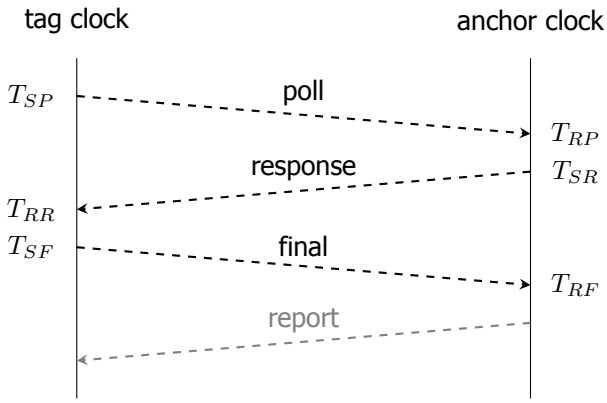


Figure 2.9: Graphical representation of the two way ranging protocol. Image recreated from [65].

2.7. Review on Range-based Mapping and Localization

The difference between range-based SLAM and range-based localization is small. Fundamentally, in range-based SLAM, the key is to estimate the locations of the beacons. Once the beacons have a known location, range-based SLAM is essentially range-based localization. The development of range-based localization, with known beacons and not formulated as a SLAM problem, is a well developed field using a wide variety of approaches to solve the problem. However, little research exists on using these *localization* tools in combination with factor graphs in SLAM.

Range-based SLAM is significantly less popular than SLAM based on LiDAR or (stereo)-cameras. Traditionally, range-based SLAM examples are found in underwater robotics where the ToF is calculated using acoustic sensors [66, 67]. Since the availability of UWB, indoor environments were also subject to range-based SLAM experiments. Many implementations are filter-based such as [68], and some exploit the ability that UWB-transceivers can also measure the range between each other [69]. Another interesting and substantially different approach, in which the range-based SLAM problem is solved using spectral sub-space identification algorithms, is shown in [70]. A critique to [70] is that it does not model the posterior, but just supplies a solution which holds no information on the underlying distribution. In [71], the range-based SLAM problem is solved using the Metropolis-Hastings algorithm, this is motivated by solving the SLAM problem using Markov Chain Monte Carlo-methods as introduced in [72]. In 2015 a graph- and range-based SLAM research focused on loop closure using UWB-waveform correlations was published [73]. Finally, a very recent publication handles the problem of biased range-measurements by finding sets of consistent measurements [74]. A 2014 review of range-based SLAM [75] observes the fact that range-based SLAM research using smoothening (factor graph representation and non-linear optimization) is incomplete.

As far as has been investigated within the scope of this thesis, no further literature concerning UWB range-based SLAM methods using the factor-graph paradigm has been found.

2.8. Summary

This chapter introduced a broad spectrum of subjects and has left out most of the conclusions that could have been drawn from them. In the coming chapters, however, many decisions will be substantiated using this information. The remaining part of this chapter will be used to refine the scope for **designing a Hybrid-Range-LiDAR SLAM (HRL-SLAM) system** and to find the common thread between these subjects.

2.8.1. Scoping the Design of a Range-based SLAM Algorithm

Indoor, robots equipped with low-cost LiDARs often fail in building a correct model of their environment. This is mainly because low-cost LiDARs in cluttered environments do not observe sufficient consistent in-

formation to effectively correct odometry errors. This problem is further extended because in that case incorrect *loop closures* are also likely to occur. The suggested approach is to correct odometry errors using range-measurements from UWB sensors rather than LiDAR observations. Range-measurements are far less susceptible to cluttered environments as it does not observe clutter, it simply returns a range.

The concept of loop-closure using UWB-sensors can be regarded in a whole different manner. Since UWB-sensors are identifiable (much as like a WiFi-SSID), it becomes trivial to close a loop once a beacon is re-discovered with confidence, identification by unique id's is very robust. This information can be used to do more robust LiDAR-based loop-closure, but it will also be shown that no explicit loop-closure needs to be done at all to correct the odometry.

For many robot-tasks, a metrically precise environment model is needed, but at the same time it is impossible to keep a metrically correct model of the whole world (or any large space) as it simply exceeds the size of our memory and is to0 cumbersome to handle. Humans tackle this problem by segmenting space and assigning semantic concepts such as rooms and hallways. Semantics are further used to create hierarchical abstractions models such as a city, a street and houses.

Semantics can be seen as a way to assign hierarchy to a spatial model: creating smaller maps that are governed by a new, simple topology is not necessarily metric. When using UWB-radios a topological map is already made: it contains landmark IDs of beacons which, in itself could also considered a holder for a submap: A beacon in that case is not only a tool to correct odometry, but also a semantic container for information.

3

A Hybrid Range-LiDAR SLAM System

This chapter lays down the skeleton of the HRL-SLAM system and motivates design choices for the algorithm based on discoveries done throughout the research. The main design-requirements can be formulated as the ability to **build a metrically consistent map in cluttered environments** and to keeping the door open to **incorporation of hierarchy and semantics to improve scalability and man-machine interaction**. The proposed SLAM-system can be seen as the composition of the following modules:

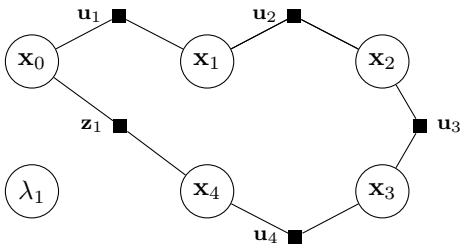
- The SLAM-processor; creates and verifies proper optimization instances.
- The SLAM back-end; responsible for optimization and inference.
- The SLAM front-end; responsible for locating beacons and feeding data the SLAM-processor.
- The grid-mapper; responsible for creating grid-maps.

3.1. Range-SLAM Formulation using Factor Graphs

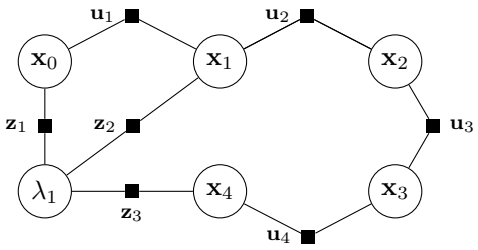
The SLAM problem is most accurately represented as a DBN [1]. This representation allows to not only assess what is right, but also on what is probable. The probabilities contain valuable information that can be used for various applications. Examples are verification of loop-closures and test whether measurements are within expectancy and can be otherwise removed. Most range-based SLAM implementations are dated and filter-based. Other approaches seen in the range-based localization scene, solve the underlying geometry problem, known as trilateration, but do not necessarily estimate the likelihood of a solution. The research area that combines range-based SLAM with factor graphs in indoor environments is small.

Basing a range-based SLAM algorithm on factor graphs can be beneficial since loop closures can happen implicitly. The downside however it that the probability distribution of a beacon-location often is multi-modal [68]. In the HRL-SLAM there are also no explicit loop-closures induced by LiDAR. Odometry-constraints are derived only from wheel encoders and IMU's on a short timespan. The error is therefore limited to that of wheel slips and some noise. The ability to correct the accumulation of odometry error lies in the fact that beacons have a fixed location in the real world and are unmistakably identifiable. This removes the need for explicit loop closure, such as scan-matching and allows implicit loop closure. This difference is graphically depicted in figure 3.1:

Another, less trivial reason to desire the graph approach, is the ability to fuse information from different sensor-sources without having to redesign the whole system. Once a factor is designed, that relates the new sensor type to a robot-pose, it can simply be connected to the graph to improve the estimate [76, 77]. Factor graphs allow a straightforward manner to fuse information that is delivered



(a) LiDAR-based loop-closure, $\mathbf{z}_1$ is the loop closure factor, but it is essentially the same as the odometry factors $\mathbf{u}$.



(b) Beacon-based loop-closure. The measurements $\mathbf{z}$ implicitly close the loop.

Figure 3.1: The fundamental difference in graph-structure between range-based and LiDAR-based graph-slam.

Variables	All robot states	$\mathbf{x}_t$,	$t = 0, \ldots, T$
	All k landmark states	λ_j,	$j = 1, \ldots, k$
	All odometry measurements	$\mathbf{u}_t$,	$t = 0, \ldots, T$
	All range-measurements	$\mathbf{z}_o$,	$o = 1, \ldots, O$
	Set of observed landmarks at time	$\mathbf{n}_t$,	$n \subseteq \{1, \ldots, k\}$, $t = 0, \ldots, T$

Table 3.1: Definition of all involved variables.

asynchronous. The graph representation also allows simple means to add meta-data to a graph which can simplify the introduction of semantics.

3.1.1. Graph Structure

In range-based SLAM the problem can be formulated as finding the solution to the optimization problem defined at 2.5:

$$\mathbf{x}^*, \lambda^* = \underset{\mathbf{x},\lambda}{\operatorname{argmax}} P\left(\mathbf{x}_{1:t}, \lambda \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}, \mathbf{n}_{1:t}\right)$$

And the meaning of each variable is defined in table 3.1:

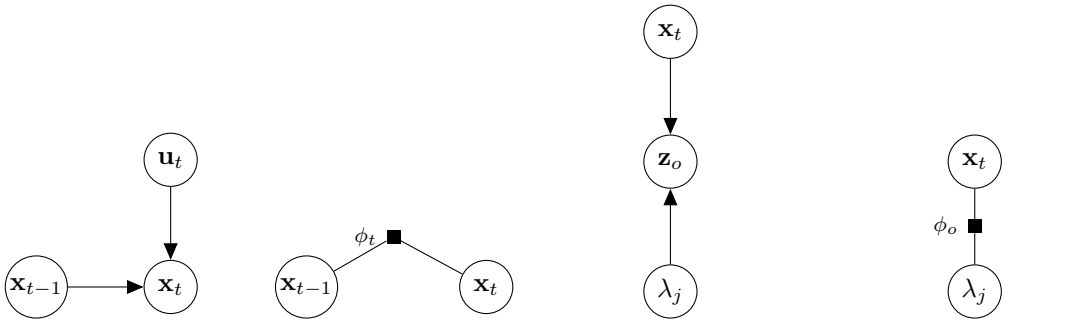
We can make a motion function that calculates the current state $\mathbf{x}_t$ of the robot based on the previous state $\mathbf{x}_{t-1}$ and the state increment that can be derived from the odometry information $\mathbf{u}_t$. Since we are working on a probabilistic approach, it is only natural to assume process noise $\mathbf{w}_t$ as well. The motion model becomes: $\mathbf{x}_t = f_t(\mathbf{x}_{t-1}, \mathbf{u}_t) + \mathbf{w}_t$ and its probabilistic representation becomes $P\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t\right)$. Analog to the motion model, we can derive a function for the observation-model: $\mathbf{z}_o = h_o(\mathbf{x}_t, \lambda_j) + \mathbf{v}_o$ and the accompanying probabilistic formulation $P\left(\mathbf{z}_o \mid \mathbf{x}_t, \lambda_j\right)$. The DBN that represents the SLAM-problem, such as the figures in 2.1, is simply a bunch of the building blocks shown in figure 3.2 concatenated.

Any DBN build by concatenating blocks from these two models results in a network in which all the dependencies between variables are defined. Because of the clear structure, the joint probability 3.1 can expressed as a product of all the separate conditionals. This transforms equation 2.6 to:

$$P\left(\mathbf{x}_{1:T}, \lambda_{1:k} \mid \mathbf{z}_{1:O}, \mathbf{u}_{1:T}, \mathbf{n}_{1:T}\right) \propto \prod_{t=1}^{T} P\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t, \mathbf{n}_t\right) \prod_{o=1}^{O} P\left(\mathbf{z}_o \mid \mathbf{x}_{t_o}, \lambda_{j_k}, \mathbf{n}_{t_o}\right) \tag{3.1}$$

To find the optimal configuration of variable-sets $\mathbf{x}^*$ and λ^* which maximize the joint probability $P\left(\mathbf{x}_{1:T}, \lambda_{1:k} \mid \mathbf{z}_{1:O}, \mathbf{u}_{1:T}, \mathbf{n}_{1:T}\right)$, which also requires setting an initial pose, a *prior* $P(\mathbf{x}_0)$, the question can be casted an optimization problem:

$$\mathbf{x}^*, \lambda^* = \underset{\mathbf{x},\lambda}{\operatorname{argmax}} P(\mathbf{x}_0) \prod_{t=1}^{T} P\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t, \mathbf{n}_t\right) \prod_{o=1}^{O} P\left(\mathbf{z}_o \mid \mathbf{x}_{t_o}, \lambda_{j_k}, \mathbf{n}_{t_o}\right) \tag{3.2}$$



(a) $P(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t)$ **(b)** $P(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t)$ **(c)** $P(\mathbf{z}_o \mid \mathbf{x}_t, \lambda_j)$ **(d)** $P(\mathbf{z}_o \mid \mathbf{x}_t, \lambda_j)$

Figure 3.2: Graphical representation of probabilistic motion and observation model. Figure 3.2a and 3.2c show DBN representations and figure 3.2b and 3.2d show the factor graph equivalent.

The argument to be maximized in 3.2 can be written as a system of equations that constrain the state-variables $\mathbf{x}$ and λ. The total size of the state is the number of poses T plus the number of landmarks k. Since the constraints depend solely on the state variables, an $N \times N$-matrix A is sufficient to describe all the possible relations between the variables. Also, each variable is constrained only locally, in a Euclidean sense: landmarks and nearby poses are within a certain *distance*. Usually this is the previous and succeeding $\mathbf{x}$-variable and the observed λ-variables. The amount of landmarks λ is in general orders of magnitude smaller then the total amount of poses $\mathbf{x}$ and even more usually only a subset is observed. Therefore, the matrix A at most holds $T \times (k+2)$ non-zero entries. Since $T \gg k$ and $k > 1$ thus $\frac{T \times (k+2)}{(k+T)^2} \approx 0$, shows that most of the entries in A are zero-entries and therefore A is sparse. This is important as it becomes beneficial to apply QR-factorization or Cholesky-decomposition which allow efficient inversion of large, sparse matrices.

3.1.2. Factors and Optimization Objective

The joint probability in equation 2.4 can be expressed as the product of $(T+O)$ constraints that each only rely on a small set of variables. The resulting computation therefore meets the description of being a large sparsely connected system, and it is a good idea to the apply the concept of factor graphs and the sum-plus algorithm for inference on the resulting network of variables. To do so, this requires the transformation of the DBN to a factor graph, which is visually illustrated for both the motion and observation model in figure 3.2b and 3.2d.

For HRL-SLAM this requires the formulation of three factors: ϕ_t, ϕ_o and a prior-factor ϕ_0.

factor	**probability**	**description**
$\phi_0(\mathbf{x}_0)$	$P(\mathbf{x}_t)$	encodes the prior information on the variable $\mathbf{x}_0$
$\phi_t(\mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{u}_t)$	$P(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t)$	describes the relation between poses given odometry information
$\phi_o(\mathbf{x}_{t_o}, \mathbf{z}_{o_j}, \lambda_j)$	$P(\mathbf{z}_o \mid \mathbf{x}_t, \lambda_j)$	describes the relation between a pose and a landmark given a range-measurement

The joint-probabilities $P(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t)$ and $P(\mathbf{z}_o \mid \mathbf{x}_t, \lambda_j)$ can be derived from the motion and measurement models by taking the likelihood [78]. If the process noise-models $\mathbf{w}_t$ and $\mathbf{v}_o$ are assumed Gaussian with respective covariances Σ_t and Σ_o, the likelihood functions, which are proportional to the probabilities $P(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t)$ and $P(\mathbf{z}_o \mid \mathbf{x}_t, \lambda_j)$ can be expressed in terms of the information matrix $\mathbf{\Omega} = \Sigma^{-1}$:

$$\phi_t = P(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t) \propto (\mathbf{x}_t - f_t(\mathbf{x}_{t-1}, \mathbf{u}_t))^\intercal \mathbf{\Omega}_t (\mathbf{x}_t - f_t(\mathbf{x}_{t-1}, \mathbf{u}_t)) \tag{3.3}$$

$$\phi_o = P(\mathbf{z}_o \mid \mathbf{x}_t, \lambda_j) \propto (\mathbf{z}_o - h_o(\mathbf{x}_t, \lambda_j))^\intercal \mathbf{\Omega}_o (\mathbf{z}_o - h_o(\mathbf{x}_t, \lambda_j)) \tag{3.4}$$

The likelihood functions can be generalized by defining the error functions $\mathbf{e}_t = (\mathbf{x}_t - f_t(\mathbf{x}_{t-1}, \mathbf{u}_t))$ and $\mathbf{e}_o = (\mathbf{z}_o - h_o(\mathbf{x}_t, \lambda_j))$. This way both the associated likelihood functions of the factors generalize to the form:

$$\phi = \mathbf{e}^\intercal \mathbf{\Omega} \mathbf{e} \tag{3.5}$$

The maximization defined in 3.2 is equals to minimizing its negative likelihood function, which can be expressed as the sum of *all* likelihood functions associated with the factors:

$$\mathbf{x}^*, \lambda^* = \underset{\mathbf{x},\lambda}{\operatorname{argmin}} \left(\sum_0^T \mathbf{e}_t \left(\mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{u}_t\right)^\intercal \mathbf{\Omega}_t \mathbf{e}_t \left(\mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{u}_t\right) + \sum_1^O \mathbf{e}_o \left(\mathbf{z}_o, \mathbf{x}_{t_o}, \lambda_j\right)^\intercal \mathbf{\Omega}_o \mathbf{e}_o \left(\mathbf{z}_o, \mathbf{x}_{t_o}, \lambda_j\right) \right) \tag{3.6}$$

The initial values for this optimization are usually simply estimates retrieved through the odometry system and range measurements. For a more detailed description of solving this non-linear set of equations the reader is referred to appendix A.

3.1.3. Calculating the Residual

To make the optimization problem computable, the residual functions need to be specified such that $\mathbf{e}_t$ and $\mathbf{e}_o$ can be computed. This requires the introduction of the models f_t and h_o and a source where we can draw values for the variables from. The information sources are tabulated in table 3.2. The

variable	source
$\mathbf{x}_t$	from the optimized set of variables, otherwise $\mathbf{x}_{t-1} + f_t\left(\mathbf{x}_{t-1}, \mathbf{u}_t\right)$
$\mathbf{x}_{t-1}$	from the optimized set of variables, otherwise $\mathbf{x}_{t-2} + f_t\left(\mathbf{x}_{t-2}, \mathbf{u}_{t-1}\right)$
$\mathbf{u}_t$	acquired through odometry system
λ_j	from the optimized set of variables, otherwise from the beacon-localization system from section 3.2
$\mathbf{z}_o$	directly from the range-sensor

Table 3.2: Source of all involved variables.

models f_t and h_o depend on the robot and used sensors. For f_t a simple motion model is used. The model f_t outputs a 2D odometry increment $\mathbf{x}_{t,t-1} = [\delta x, \delta y, \delta\theta]$ given the pose $\mathbf{x}_{t-1}$ ($[x, y, \theta]$) and a control input $\mathbf{u}_t$ ($[\delta x, \delta y, \delta\theta]$).

The model h_o returns an expected range-measurement given the robot's state $\mathbf{x}_{t_o}$ (the state at the time of the observation $\mathbf{z}_o$) and the location of the observed beacon λ_j. Note that there need to be at least three measurements to each λ_j, otherwise the problem is ambiguous [79].

In the case of h_o, the function returns a scalar that describes the Euclidean distance between a pose and a point. In the case of f_t, however, the function returns a relative transformation between the poses $\mathbf{x}_t$ and $\mathbf{x}_{t-1}$ which is not Euclidean: the transformation consists out of a translation, which is Euclidean, but also out of a rotation from the rotation group $SO(2)$ which is not Euclidean. Optimizing the relative error between consecutive poses using a Euclidean residual error-function is bound to give poor or wrong results. It is therefore common to perform the least-squares optimization on a manifold instead of in Euclidean space. The reader is referred to appendix A for the derivation of an appropriate error-function for the 2D rigid-body transformation residuals. The models f_t and h_t often are non-linear. In such a case it is common to use a first-order Taylor expansion: $f_t\left(\mathbf{x_t}, \Delta\mathbf{x}_t\right) \approx J_t\Delta\mathbf{x} + f_{t-1}(\mathbf{x_{t-1}}, \Delta\mathbf{x}_{t-1})$

3.1.4. Suitable Estimator Functions

The ability to calculate the maximum likelihood is possible under the assumption that the underlying error distributions of f_t and h_t are Gaussian. This is a naive assumption. In this section the use of a Gaussian error-model for pose-constrains is justified and a more elaborate approach for the range-constraints is explained.

In HRL-SLAM odometry factors only connect poses that are both spatially and temporally close to each other. This makes the assumption of an unimodal Gaussian error distribution more appropriate as

the largest outliers are still within the size of a small odometry update. This assumption is significantly more acceptable compared to the Gaussian assumption for loop-closure error models. Explicit loop-closure, however, does not exist in HRL-SLAM. This means a least-squares error function should suffice in the optimization.

For the range-model, especially using TDoA measurements, requires a more informed decision. First a differentiation should be made between two different magnitudes of errors that can occur within range-based SLAM. A set of range-measurements in combination with the accompanying poses can create a multi-modal distribution for a beacon-location in which the modes of all possible locations are not overlapping, an example situation is shown in figure 3.3. In this case, the difference between the modes usually is significantly large that it could introduce an unrecoverable convergence to a local minimum in the optimization process if the distribution converges to the wrong mode. The second error type is natural to TDoA range-measurements using UWB-radios. This is generally modeled as a sum of a Gaussian component and a Bernoulli-distribution that accounts for the effects of possible Non Line of Sight (NLOS) signals [80].

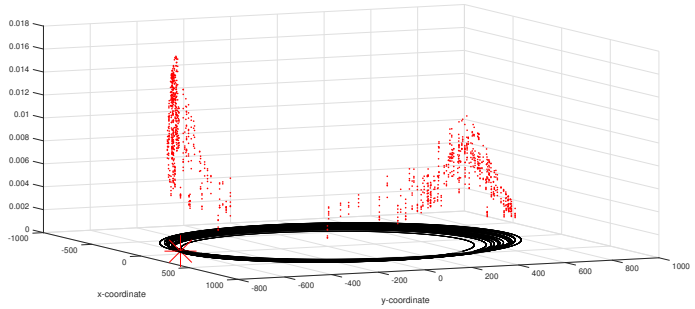


Figure 3.3: An example of a (discrete) multi-modal probability distribution for a beacon-location (the red asterisk depicts the true location) given a set of poses $\mathbf{x}$ and range-measurements (black circles in the xy-plane have a pose as circle-center and a radius equal to the range-measurement. The red dots are probable beacon-locations based on a circle-intersection model (explained in section 3.2) and they are assigned a weight using the residual of the trilateration equation of 3.5, which is illustrated along the z-axis. Note that the distribution is clearly multi-modal and that the residuals of the trilateration equation can not be used robustly as a single decision aid.

Coping with a multi-modal probability distribution in the optimization procedure directly can be difficult. In trilateration there are often just a few peaks and most often just two peaks, as illustrated in figure 3.7. Such distributions can be approximated by a mixture of Gaussians. This mixture can be included in the optimization using a max-mixture approach as described in [37]. The challenge lies into designing a Gaussian for each failure mode, this will also require calibrating the sum of Gaussians as the NLOS-probability differs per environment.

Opposed to embedding the modality-reduction of the beacon-location distribution in the optimization, the multi-modality can also be reduced as a foregoing step before adding the beacon-location and associated factors to the optimization. This way, in the optimization procedure we are only concerned with TDoA-related noise and treat the multi-modal problem as a stand-alone problem. In section 3.2 a beacon-localization algorithm is introduced based on [67]. It is used to create an initial guess for a beacon-location that at least estimates the correct mode. For the remaining TDoA error the Gaussian assumption is more appropriate as we reduce the modality of the distribution using the method described in the next section.

For the remaining part we consider two remaining error-models for the range-measurements: typical TDoA-errors such as described in [80] *and* the possibility that the initialization of the beacon-location is *wrong*. UWB-measurements in general are accurate and the difference in mode-peaks for TDoA characteristics are usually so small that an approximation using a unimodal Gaussian often works sufficiently [81]. To cope with the error introduced in the case of a wrongly initialized beacon, e.g. to a wrong location in which the optimization converges to a local minimum, Tukey's biweight cost function is used.

This weight function acts much the least-square estimator for small errors, but the penalty becomes constant beyond a certain error. This approach is motivated by that a wrongly initialized beacon is certainly going to produce outliers and Tukey's biweight cost function will *limit* the effect of the outliers that would otherwise in the case of a least-square estimator could potentially harm the whole optimization.

3.1.5. Incremental Optimization

The quality of the initial estimate increases that odds of successfully performing non-linear optimization greatly. If the optimization would first be constructed as a whole, and the raw-odometry is used as the initial value for each variable, the odds are high of getting stuck in a local minimum [82]. One way to improving the initial estimate is by solving a linear approximation of the SLAM-problem as done in [28, 29]. Since the latter is rather computational intense as the whole SLAM-problem, even though linearly approximated, is calculated on each iteration. Therefore, the choice is made to implement the rather traditional approach of incrementally adding odometry increments on top of the optimized recent pose. Specifically, the iSAM2 algorithm as described in [8] is used, as it also automatically takes care of maintaining sparsity in by efficient column-reordering and does efficient Jacobian re-linearization. For the non-linear optimization Powell's Dog-Leg method is picked as it is computationally more efficient compared the Levenberg-Marquardt without compromising significantly on quality [26].

3.1.6. Dynamical Spatial Constraints: A Moving Prior

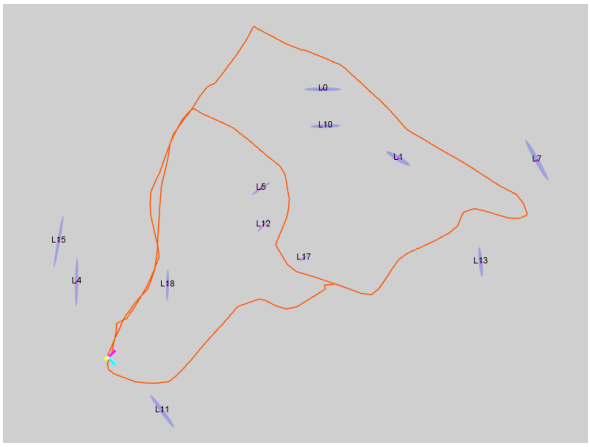
As mentioned in section 3.1 a prior needs to be added to make the system of equations determined. Fundamentally, without the prior-factor (or unary-factor) we have a constraint-satisfaction problem. Adding the prior-factor transforms the constraint-satisfaction problem into an constraint-optimization. Without a prior the system can not be solved. A prior is an odd duck as it incorporates information known previously to starting the mapping procedure, which is in general nothing. But since a system can not be solved without, a prior is usually added as a constraint $\mathbf{x}_0 = \begin{bmatrix} x = 0 & y = 0 & \theta = 0 \end{bmatrix}$ with a very low uncertainty. While this assumption does little harm to the optimization, except for constraining the first pose to a fixed pose, it does add a misleading influence on the marginals (covariance of a pose). Adding a very uncertain prior-constraint on the first pose adds too much uncertainty too all consecutive poses, adding a very small uncertainty will make all the covariances dependent on how far it is away from the first pose. In HRL-SLAM the prior-constraint with a high certainty is put on the most recent pose, instead of the first pose. This also makes sense as all uncertainty is then calculated relative to the most recent pose, which to some extent is the *most certain* pose. This way the uncertainties are estimated in the robot's current local frame, which is more usable. This difference is depicted in figure 3.4.

3.2. Scalable Beacon Discovery

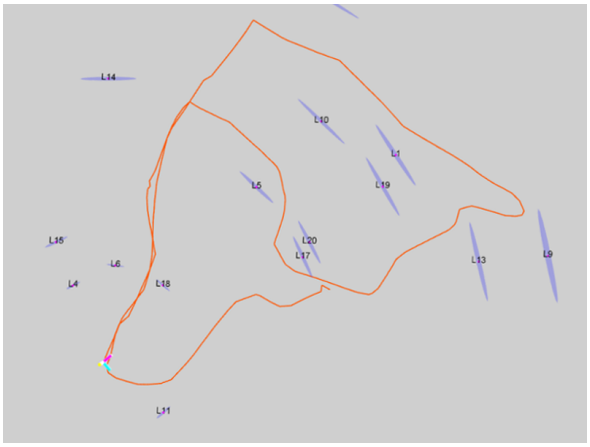
The positions of all k beacons, $\lambda_j, \forall j \in \{1, \ldots, k\}$ are, just as the robot poses $\mathbf{x}_{0:T}$, variables in the non-linear optimization process. For each beacon an initial estimate must be given before the associated constraints can be used for pose-correction. Due to the ambiguities that can arise from range-only measurements, since range-only measurements only predict the pose-location and not the pose-rotation, an initial estimate for λ_j that is far off the true value can result in an inescapable local minimum in the optimization procedure.

A range measurement is a scalar number that describes the Euclidean distance between two points (or a pose and a point, or two poses). If both points are known, the probability of the range-measurement can be assessed. In range-based SLAM however, locations of the beacons are initially unknown. The initial task therefore becomes finding the locations of the beacons. Finding the location of a beacon, given a set of range-measurements at known locations is called *trilateration* and mathematically can be described in an $Ax = b$-form as seen in equation-set 3.5. The trilateration problem is a simple geometry problem if the locations at which the range measurements occur are *not* collinear*.

*Three or more points are collinear if the lie on the same line



(a) Covariances with a prior on $\mathbf{x}_0$.



(b) Covariances with a prior on $\mathbf{x}_T$, the most recent pose.

Figure 3.4: Difference between constraining the first pose or the most recent pose. It can be seen in that in figure 3.4b the marginals are centered around the initial pose $\mathbf{x}_0$, which is near λ_{17}. In figure 3.4b the marginals are relative the the last pose (the green-red-blue axis).

$$\begin{aligned}
(\lambda_x - \mathbf{x}_1)^2 + (\lambda_y - \mathbf{y}_1)^2 &= d_1^2 \\
(\lambda_x - \mathbf{x}_2)^2 + (\lambda_y - \mathbf{y}_2)^2 &= d_2^2 \\
&\vdots \\
(\lambda_x - \mathbf{x}_{T-1})^2 + (\lambda_y - \mathbf{y}_{T-1})^2 &= d_{T-1}^2 \\
(\lambda_x - \mathbf{x}_T)^2 + (\lambda_y - \mathbf{y}_T)^2 &= d_T^2
\end{aligned} \tag{3.7}$$

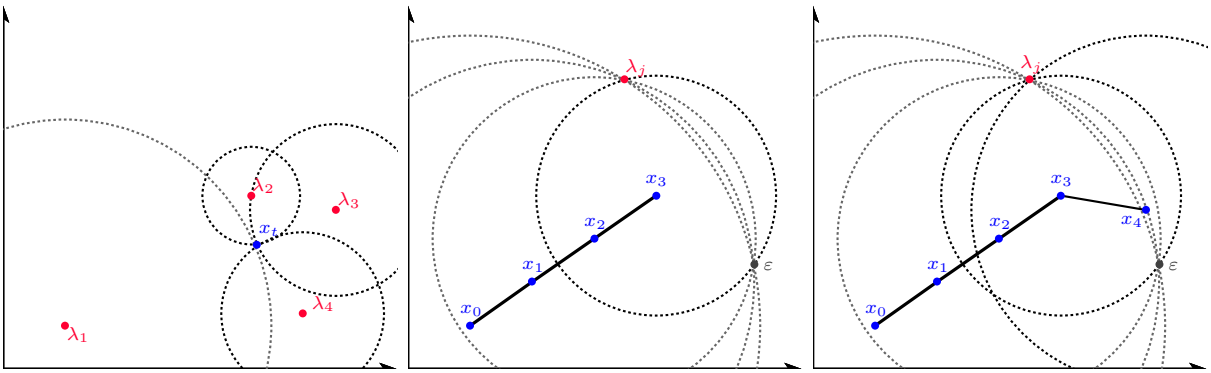
$$\begin{aligned}
\lambda_x^2 + \lambda_y^2 - 2\mathbf{x}_1\lambda_x - 2\mathbf{y}_1\lambda_y &= d_1^2 - \mathbf{x}_1^2 - \mathbf{y}_1^2 \\
\lambda_x^2 + \lambda_y^2 - 2\mathbf{x}_2\lambda_x - 2\mathbf{y}_2\lambda_y &= d_2^2 - \mathbf{x}_2^2 - \mathbf{y}_2^2 \\
&\vdots \\
\lambda_x^2 + \lambda_y^2 - 2\mathbf{x}_{T-1}\lambda_x - 2\mathbf{y}_{T-1}\lambda_y &= d_{T-1}^2 - \mathbf{x}_{T-1}^2 - \mathbf{y}_{T-1}^2 \\
\lambda_x^2 + \lambda_y^2 - 2\mathbf{x}_T\lambda_x - 2\mathbf{y}_T\lambda_y &= d_T^2 - \mathbf{x}_T^2 - \mathbf{y}_T^2
\end{aligned} \tag{3.8}$$

$$\begin{bmatrix} 1 & -2\mathbf{x}_1 & -2\mathbf{y}_1 \\ 1 & -2\mathbf{x}_2 & -2\mathbf{y}_2 \\ \vdots & \vdots & \vdots \\ 1 & -2\mathbf{x}_{T-1} & -2\mathbf{y}_{T-1} \\ 1 & -2\mathbf{x}_T & -2\mathbf{y}_T \end{bmatrix} \begin{bmatrix} \lambda_x^2 + \lambda_y^2 \\ \lambda_x \\ \lambda_y \end{bmatrix} = \begin{bmatrix} d_1^2 - \mathbf{x}_1^2 - \mathbf{y}_1^2 \\ d_2^2 - \mathbf{x}_2^2 - \mathbf{y}_2^2 \\ \vdots \\ d_{T-1}^2 - \mathbf{x}_{T-1}^2 - \mathbf{y}_{T-1}^2 \\ d_T^2 - \mathbf{x}_T^2 - \mathbf{y}_T^2 \end{bmatrix} \tag{3.9}$$

Figure 3.5: The trilateration problem rewritten from a set of constraints to an $Ax = b$-form, which can be solved for x using the pseudo-inverse. $\lambda_{x,y}$ is the location of the beacon, $\mathbf{x}_t$ and $\mathbf{y}_t$ are the x and y coordinates of the robot in a specific frame at time t. d_t is the range-measurement at time t.

In such a case the pseudo-inverse returns reliable initial estimate. In the case of localizing a robot by using multiple beacons, the measurement seldom lie on the same straight line since the beacons are usually not placed on the same line. The common robot-localization situation is depicted in 3.6a. However, once the situation is inverted: find the beacon-location based on a series of range-measurements taken from the robot, the collinearity of measurement-locations becomes a problem when using the pseudo-inverse. This is illustrated in figure 3.6b. Pose-trajectories often approximate a straight line on short distances and the measurement locations become notably more uncertain once the path devi-

ates from straight lines. A slight deviation from the collinear path will destroy the ambiguous situation, illustrated in figure 3.6c. The situation of beacon-localization therefore requires a robust approach.



(a) A typical robot localization situation using multiple beacons, the robot location is unambiguous.

(b) A collinear pose-trajectory creates an ambiguous situation. λ_j and ε are equally likely locations for the beacon-location.

(c) A non-collinear pose-trajectory kills the ambiguity.

Figure 3.6: Illustration of the effect of a collinear measurement locations. The dashed circles represent the circles with a radius equal to the range-measurement. Figure 3.6a depicts the case in which multiple beacons are used to estimate a robot pose x_t. In figure 3.6b and 3.6c the situation is shown when a robot tries to estimate the location of a beacon. Often, the robot path approximates a straight line, introducing collinearity to the problem. Assuming the wrong position for a beacon, to in this case ε, will very likely result in a sub-optimal configuration from which the non-linear optimization process can not recover. Figure 3.6c shows how deviation from the collinear pose-trajectory introduces the preference for, in this case, the solution λ_j. The deviation, however, could also have been the result of measurement noise.

3.2.1. A Stochastic Beacon-Localization Algorithm

In the previous section it was shown that beacon-localization requires a robust method because the cost for wrongly initializing a beacon is high, as it results in a local-optimum. By robust is meant that no decision should be made if the robot is in a situation such as in figure 3.6b but also that it only makes a decision on a beacon-location once it is confident that the deviation of collinear situation is *not* caused by process-noise (which could be caused by odometry or the range-measurements itself).

In HRL-SLAM a voting scheme in combination with a *spectral analysis*-aided decision mechanism is used to find the locations of the beacons. This approach shares a lot of commodities with the spectral filter introduced in [67]. It is modified such that the algorithm becomes scalable and performs better. Also the need for the spectral filter to be ran *online* is removed as the algorithm is only used for beacon-localization. The filtration of the measurements through Kalman-filtering is replaced by the smoothening of variables that results from the optimization procedure. This makes the beacon-localization mechanism significantly more robust as now beacons can also be discovered on loop closure.

3.2.2. Voting

A discrete probability distribution describing the location λ_j given a set of N measurement locations $\mathbf{x}_{t_o}$ and N associated range-measurements $\mathbf{z}_o$ can be constructed by finding all the intersections between the circles that have $\mathbf{x}_{t_o}$ as center and have radius $\mathbf{z}_o$. One could also sample random points from the circles, but since the interest lies in the intersections it is reasonable to sample the two points of intersection per range-measurement pair (or none if the do not intersect) opposed to sampling many points from the whole circle-set.

All the computations are done in a frame relative to a pose. For simplicity the calculations are done in the frame of $\mathbf{x}_0$, indicating that $\mathbf{x}_0 = [0, 0, 0]$, but it could be any frame. For each pose-set, the points of intersections are calculated (if there are intersections, otherwise nothing happens) and are plotted on a discretized map. Each time an intersection is put into a grid-cell, the value of that cell, and neighboring cells are incremented with 1. The increments added to the neighboring cells are to account for flaws that arise from the discretization of placing points with continuous coordinates in a grid-map. The grid-cell with the highest hit-count can then be considered as the most likely location for λ_j. In order to gain confidence in the winning peak the constraint is added that the winning peak should at least have twice as many hits as the second biggest hit. This requirement is copied from [67].

To improve the performance of the voting scheme, the values of the robot poses are drawn from the optimized set of robot-poses from the SLAM optimization. This is different from many approaches as they assume dead-reckoning (raw-odometry) is locally accurate enough to estimate the beacon-location. While this is not necessarily untrue, it makes it impossible to locate beacons that have initially failed to be located and are observed again after a while: the raw odometry over a long path is erroneous per definition and yields very inconsistent circle-intersections. See an example in figure 3.7. If a coarser cell-size is used, the beacon-location can be approximated also in the raw-odometry case. However, a coarser estimation is dangerous for the optimization, especially if the translational component of the motion increments are roughly the same as the change in range measurements, e.g. the robot is moving towards, or away from, the beacon in a straight line. So, to get an accurate beacon-location estimate, the interest lies in minimizing the cell-size while keeping the grid-map computable *and* avoiding the hit-distribution to approach a uniform distribution in which the voting scheme does not work.

3.2.3. Spectral Filtering

In [67] a spectral filter was used to remove outliers that were consistent with each other, a situation that arises commonly in underwater using acoustic sonar. The approach is based on finding the largest set of consistent measurements and base the assumption solely on the set of consistent measurements. One of the main flaws in the voting-mechanism is that there is a trade off between certainty and accuracy. A small grid-cell size decreases the odds of ever having a peak that is significantly larger than the second biggest peak, a large grid-cell size, however, increases the odds initializing the beacon wrongly, resulting in problem for the non-linear optimization part of the SLAM-algorithm.

The spectral filter can be used to create a small subset of the intersections such that only intersections that carry a certain consistency. Two measurements are consistent if they can both be explained by a beacon-location. These consistencies are then formed into an undirected graph per pair. In this graph the vertices are measurements and consistent measurements are connected by an edge. Separating the set of consistent measurements from the set of inconsistent measurements can now be casted as a discrete optimization problem for which want to remove the minimal amount of edges such that there are two disconnected sets. This problem can be solved using the Min-max cut algorithm proposed in [83].

Consider the set of range measurements, $M_i : 1 \leq i \leq N$. And a $N \times N$ adjacency matrix A is created using:

$$A_{ij} = \begin{cases} 1 & i \neq j, M_i \quad \text{and} \quad M_j \quad \text{are consistent} \\ 0 & \text{otherwise} \end{cases}$$

Now take u to be an $N \times 1$ *indicator* vector of which each element either is 0 or 1 and define $u_i = 1$ if measurement M_i is an inlier and $u_i = 0$ otherwise. The quality of a cut can now be measured with $r(u)$, which is a scalar function of u:

$$r(u) = \frac{u^\intercal A u}{u^\intercal u} \tag{3.10}$$

$u^\intercal A u$ now is twice the number of edges within the inlier cluster and $u^\intercal u$ is the total number of vertices's that is classified as inlier. Finding $\operatorname{argmax}_u r(u)$ is not trivial since u has discrete entries: either one

or zero. Common in graph-partitioning, the problem can be relaxed by allowing u to hold continuous variables [84] and now $\text{argmax}_u\, r(u)$ becomes an Eigenvalue problem of the form $Au = \lambda u$. Using the eigenvector associated with the largest eigenvalue λ gives us the optimal $r(u)_c$, in a continuous form. The remaining task is to discretize u again, using a fixed threshold θ:

$$u(i) = \begin{cases} 1 & u(i) > \theta \\ 0 & \text{otherwise} \end{cases}$$

How well the u_d describes u_c can be assessed by maximizing $u_d \cdot u_c$ as shown in [67]. Finding an optimal value for θ can only be found through a brute-force search, calculating the dot-product for each θ.

Remarks on Scalability of the Spectral Filter

There are two aspects that make this algorithm scale badly: A grows quadratically as more measurements flow in and the complexity of calculating the Eigenvalues is, in general, $\mathcal{O}\left(n^{2.376}\right)$ [85] (n is the amount of measurements). Secondly, finding θ is done by calculating $u_d \cdot u_c$ for each suggested θ - which is feasible as long as we limit the amount of testable thresholds *and* the length of u is still workable.

To fix the scalability issues a simple relaxation is added to the spectral filter: instead of operating on the whole set of measurements, a fixed amount of random samples are drawn from the measurement set. A smarter approach could also be used, such as removing measurements that appear as outliers. However, it should be taken into consideration that this usually introduces new computations while we are in the business of making it *less* computationally intensive.

3.3. Generating Grid-maps

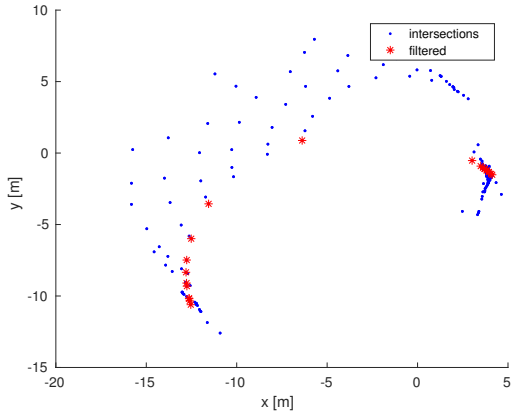
One of the most commonly used environment models is the occupancy grid-map. These maps are usually build by aligning scans with previous scans. HRL-SLAM has a very simple grid-map mechanism. For each LiDAR observation the transform $\mathbf{v}_o$ to the most recent optimized variable $\mathbf{x}_t$, is put in a database along with the recorded scan. This way each LiDAR observation that is stored in the database has a transform to a robot-pose, which in itself is also a transform to the map frame. Using $\mathbf{x}_t \oplus \mathbf{v}_o$ transforms the LiDAR scan to a global map frame. This only gives us an indication of which cells are occupied and not which are unoccupied. The transforms $\mathbf{v}$ are derived from the raw-odometry. The distance traveled since the most recent optimized variables is usually small, as in less then a meter (a typical low-cost LiDAR operates at $10\,\text{Hz}$), this is decent approximation. Effort could be put into matching scans to enhance the performance. Figure 3.8 shows how the LiDAR database is connected to the factor graph network.

Implementing routines for also approximating the unoccupied cells is necessary for building a usable occupancy grid-map. However, for this research, plotting solely the LiDAR-beam end-points is sufficient for demonstrating the grid-mapping capabilities of using range-corrected pose-trajectories in combination with LiDAR observations.

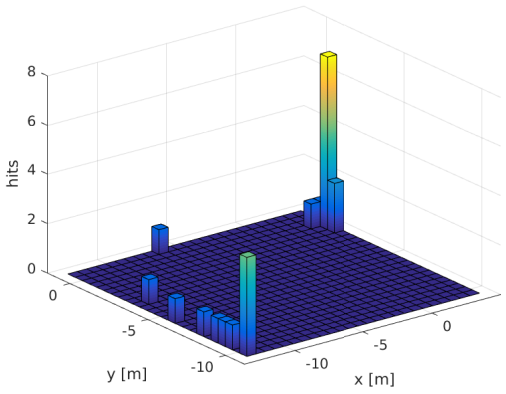
3.4. Scalable Localization

Often, the situation arises that a robot loses localization. In such a case, non-linear optimization is not a very reliable tool for localization as there is no decent initial value. In such a case, MCL can be used to retrieve the position of the robot. MCL needs a model of the environment so it can evaluate the sensor-observations in combination with the evolutions of state-hypotheses. The number of hypotheses, the size of the particle population, and the time it takes to converge to a solution is depended on the size of the environment-model.

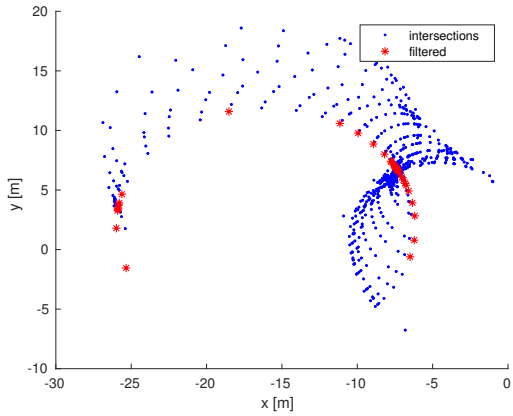
Using the UWB-radios, the environment model can be partitioned into smaller pieces: groups of observations that were within the neighborhood of an UWB-beacon. This information can be used be



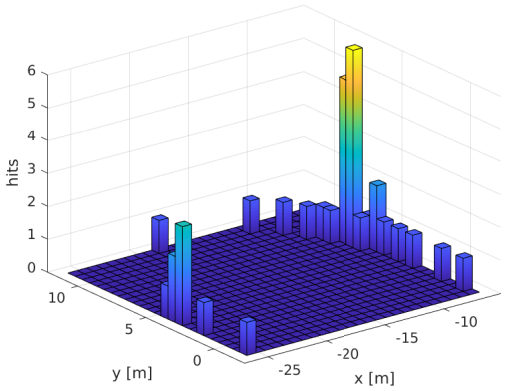
(a) Point cloud of all intersections.



(b) Histogram of the filtered set of intersections.



(c) Point cloud of all intersections.



(d) Histogram of the filtered set of intersections.

Figure 3.7: The top-row and bottom-row each represent the situation of initially failing to locate a beacon and re-encountering the beacon after a loop. The top-row represents the case in which the pose variables are extracted from the optimized set of variables, in this case the beacon is located. The bottom-row shows the case of using the raw-odometry, in which beacon-localization fails. In figures 3.7a and 3.7c the red asterisks depict the filtered set of intersections as explained in 3.2.3.

to create a much smaller model for the MCL scheme. Because HRL-SLAM is a factor graph-based, the factors store the information which variables were linked to specific beacons. Using the graph, and a definition for a neighborhood a subset of the variables and factors can be retrieved to create a small map. In this thesis the neighborhood definition is kept simple yet effective: given the observation of beacon λ_j, retrieve all variables and related factors that are within a $15\,\mathrm{m}$ radius. This created subgraph can also be used to derive a grid-map as described in section 3.3. It should be noted that the current neighborhood function is dependent on a range parameter, which is set to the range of the UWB-radios. This parameter can be set arbitrarily.

3.5. Summary

This chapter introduced a SLAM-system that uses range-measurements to estimate beacon-locations and subsequently corrects the pose-trajectory. The LiDAR's purpose is limited to solely to plotting observations in a grid-map opposed to also contributing to pose-correction. The beacon-localization algorithm is based on Olson's approach described in [67] and is extended with a subtle fix that should allow on-line computation in indoor environments. Secondly, opposed to Olson's approach which used a Kalman-filter in combination with the beacon-localization algorithm, the beacon-localization algorithm

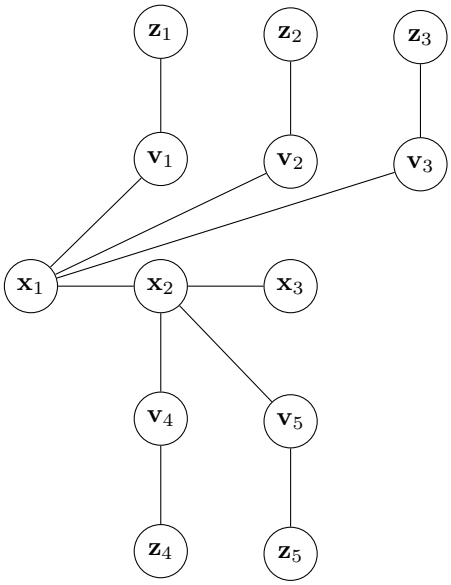


Figure 3.8: This scheme denotes the structure of the LiDAR database and how it is connected to the variable-set in the factor graph. Note that the transforms $\mathbf{v}$ are derived directly from the raw odometry.

is used in combination with a factor graph approach. This brings the great advantage that poses that are estimated prior to beacon-localization can now also be corrected by the smoothening-effect that comes from the non-linear optimization. In Olson's Kalman-filter approach it is acknowledged that the first part of the path is clearly wrong and is not corrected, even though the information is available. This is because the Kalman-filter does not re-linearize Jacobians, and as a consequence, errors are hard-coded into covariance matrices. Using the factor graph approach, measurements can be stored and used once a beacon is estimated. In this case, the affected Jacobians are re-linearized given the new information and such, poses from the past are also corrected. This contributes significantly to building a metrically consistent map.

The factor graph representation used to model the SLAM-problem has proven itself to be a golden-standard, mainly for its smoothening capabilities. While other benefits of factor graph-SLAM mentioned in literature are often described in terms of computational efficiency, the applicability of belief-propagation algorithms and efficient non-linear optimization, the incorporation of hierarchy and semantics into SLAM is a seemingly unmentioned benefit. In HRL-SLAM the graph-structure and clear separation of pose and beacon variables is used to partition the graph into local clusters. The factor graph has a clear topology, and HRL-SLAM takes advantage of that for the creation of local maps. The local-map implementation in HRL-SLAM is of the simplest kind. A local-map implementation can be based on factor-connectivity, but the graph can also be partitioned using interesting graph-partitioning techniques [84] or maybe the pose-variables can be assigned meta-data and be used in a generic clustering approach. Either way, a graph allows a far richer tool-set to create hierarchies opposed to fusing information in a grid-map.

4

Experiments

In this chapter two different types of experiments are introduced that were designed to assess the performance of the proposed HRL-SLAM algorithm. Each type consists out of two different environments. The first type of experiments are experiments that were done in simulation and are called the *Maze* experiments. The second type of experiments are based on data from a real robot. The real data consists out of two commonly used range-dataset known as the *Plaza* datasets, supplied by [12]. The chapter is divided in three parts. The first part introduces the datasets and introduces the used noise-models for simulation. The second part introduces the actual experiments. The chapter ends with a proposal for a metric assessing the performance.

The goal of the experiments is to demonstrate that the HRL-SLAM algorithm can create usable environment models. These models can be measured visually by inspecting grid-maps and metrically by analyzing the pose- and beacon-location errors.

While the simulation experiments allow us to vary model-parameters easily, the Plaza datasets allow verification and comparison. A real experiment that uses UWB-radios and a low-cost LiDAR is absent. Appendix C describes the work that has been done towards setting up such an experiment using using LEA, the robotic stroller developed at RCS.

4.1. Datasets

In total, for this research, there are four different datasets of which two are synthetic and are acquired through simulation. The two latter datasets are the Plaza 1 and Plaza 2 dataset. The simulation experiments have LiDAR-data while the Plaza experiments do not have LiDAR-data, this limits the ability to generate grid-maps for those experiments. The experiments using the synthetic data, Maze25x25 and Maze75x75, are repeatable and the noise is varied per experiment. This is important as the SLAM-algorithms which are benchmarked on just a few real-data sets, lack statistical certainty of the success-rate. Successful convergence in SLAM is not as consistent as sometimes is presented [82]: tuning an algorithm can often lead to successful convergence but it can just as easily add instability, and therefor many results on real-data should be taken with a grain of salt. The Maze experiments are repeated with different noise levels for a multitude of times to validate the consistency of the mapping performance. The experiments using the Plaza 1 and Plaza 2 dataset, obviously, have fixed data, but are used to compare results with previous work on those datasets. The simulation experiments have been performed in V-REP [11], which is a physics simulator specifically designed for robotic simulations. The four datasets are tabulated in table 4.1.

In the Maze experiments, a model of a Pioneer3DX, shown in figure 4.1a, a common differential drive-robot used in many scientific publications, is used. It is equipped with short-range LiDAR that mimics the performance of a RPLiDAR: a 270 degree view at a resolution of one degree and a maximum range of 6 m. It is also equipped with an UWB-radio with an adjustable range. By adjustable it is meant

dataset	trajectory length	measurements	ground-truth	grid-map
Maze 25x25	194 m	±500	available	yes
Maze 75x75	621 m	±1500	available	yes
Plaza 1	1860 m	3529	available	no
Plaza 2	1354 m	1816	available	no

Table 4.1: The four different datasets. The simulation datasets will be repeated for the same ground-truth trajectories but with different noise, this affects the range- and odometry-measurements effectively rendering each experiment unique. The Plaza datasets will be used to verify the algorithm on real data and will be used as comparison to other algorithms on the same datasets. **Dataset** refers to the name of the dataset. **Trajectory length** refers to the length of the ground-truth path. **Measurements** indicates how many range-measurements are available. **Grid-map** indicates whether grid-maps have been produced on-line in the experiment.

that it can be changed as a simulation parameter. In the Maze datasets the maximum range for UWB-measurements is conservative: up to a maximum of 10 m in Maze 25x25 simulations and 25 m in Maze 75x75 experiments. In the Plaza experiments, an autonomous mower, shown in figure 4.1b, was used to create the Plaza datasets. The mower is a 3-wheeled robot of which the two front wheels are the drive wheels while the back wheel steers. It has an UWB-radio from Multispectral Solutions that can measure up 80 m [12].

(a) A screenshot from the Pioneer3DX in V-REP.

(b) The autonomous lawn mower equipped with an ultra wideband-sensor. Photo from [12].

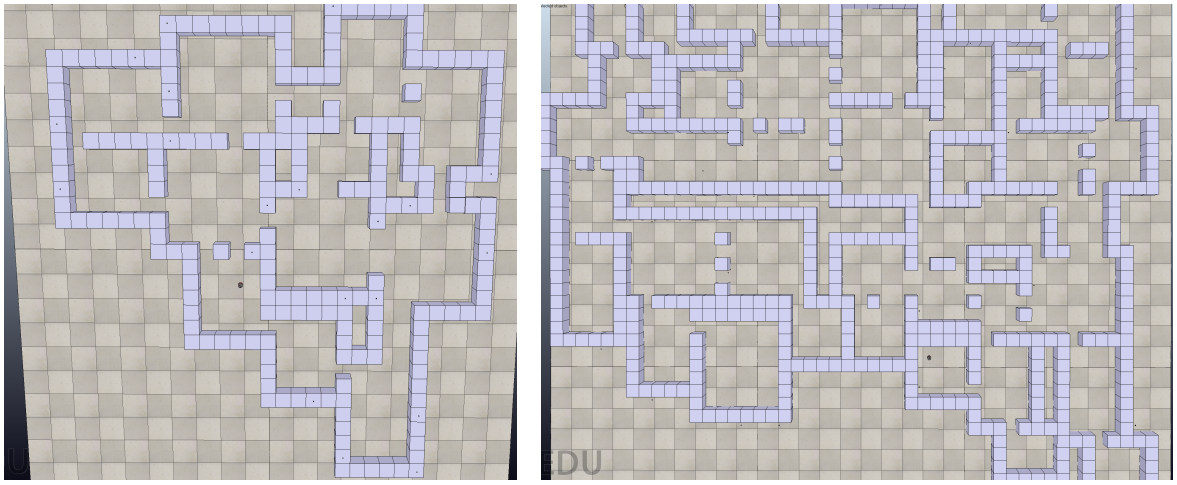
Figure 4.1: The two robots used in the four experiments.

4.1.1. The two Maze environments

In the VREP-environment two mazes have been build. The mazes were were randomly created using a maze-generator inspired by [86]. Maze 25x25 is based in an environment that covers 625 m^2, Maze 75x75 is based in an environment that covers 5625 m^2. Maze 25x25 contains twenty randomly placed short-range (10 m) UWB-beacons, Maze 75x75 also contains twenty randomly placed beacons but their range can be varied up to maximal of 25 m. All beacons in the environment, also the UWB-radio on the robot itself, are placed at the same height. This way the trilateration is a two-dimensional problem.

4.1.2. Plaza environment

The Plaza datasets have been created using an autonomous lawn-mowner as shown in figure 4.1b with an UWB-radio. The experiments took place in the plaza of Robot City at Carnegie Mellon University. Also in both the Plaza datasets, the height of the UWB-radios on the robot was the same as the four other radios that were placed in the plaza. Again removing the necessity of doing the trilateration in 3D.



(a) A screenshot of the Maze 25x25 environment.

(b) A screenshot of the Maze 75x75 environment.

Figure 4.2: The two different mazes, Maze 25x25 covers 625 m^2 and Maze 75x75 covers 5625 m^2. The black dot in both figures is the Pioneer 3DX robot at its start location.

4.1.3. Ground Truth Trajectories

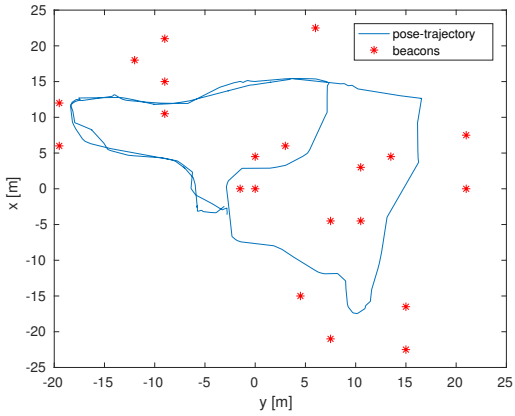
In this section the ground truth-trajectories are shown. The ground truth-trajectories show the *true* traveled path of the robot. In the case on the Maze datasets this is *the absolute* ground truth to which process noise has been added to create a noisy odometry measurements. In the case of the Plaza datasets the ground truth has been established using GPS and it is supposedly accurate up to two centimeters [12].

4.2. Simulating Noise

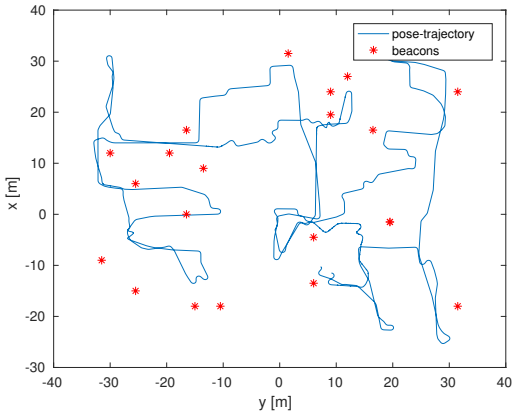
In the Plaza datasets the noise is created by the actual physical sensors and the challenge lies in creating the ground truth. In simulation, and therefore our Maze datasets, the ground truth is readily available and process noise needs to be added to the *perfect* range- and odometry-measurements in order to resemble the output of real sensors. The advantage of simulation is that experiments can be repeated with different additive noise on each run. This contributes to statistically demonstrating the robustness of the algorithm. The repeated experiments strategy is effectively the same as the Monte Carlo cross-validation method used in [82] on Olson's Manhattan Dataset, shown in 2.2b.

4.2.1. Ultra-wideband Noise

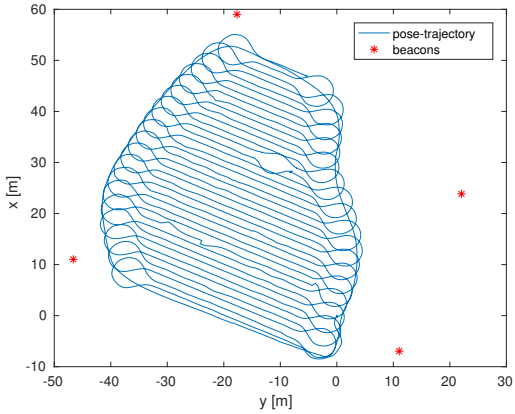
The UWB-sensors return a Euclidean distance between two UWB-radios, also often called beacons. The sensors are assumed to work in a 360 degree up to a defined range, as also defined in the Decawave DWM1000 data-sheet [87]. From experimentations done using the DWM1000-system at RCS the maximum range for reliable ranging, indoor, is between 20 m and 25 m at at least 2 Hz. Note that reliable in this case refers to successfully executing the range-protocol as defined in figure 2.9. Accuracy refers to whether the measured distance describes the true distance. In [88] research on UWB-noise-distributions was done and the results are tabulated in table 4.2. In general, due to the lack of long tails, the distributions are roughly symmetric and can be approximated by a Gaussian distribution with a small positive bias. From [88] conclusions are also drawn on NLOS situations but these effects are neglected for the simulation. In general, UWB is assumed to be able to penetrate common indoor materials [88, 89] without affecting the range dramatically. The long range measurement done in the Plaza experiments experience more severe biases at long ranges [12], these effects are also ignored because only short range measurements are used in the Maze experiments.



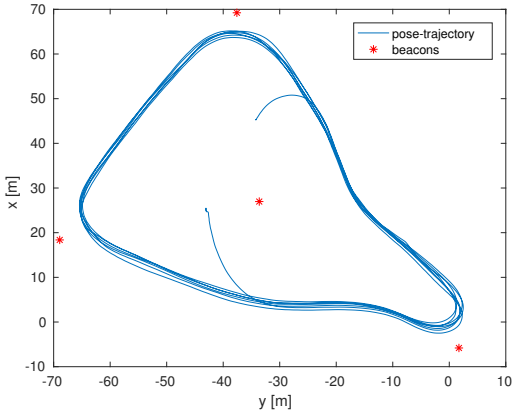
(a) Ground truth in the Maze 25x25 dataset.



(b) Ground truth in the Maze 75x75 dataset.



(c) Ground truth in the Plaza 1 dataset.



(d) Ground truth in the Plaza 2 dataset

Figure 4.3: The ground truth trajectories for all four experiments. The ground truth refers to the *true*-trajectory and the error is defined as the difference between the calculated path and the ground truth.

In the VREP-simulation a range-measurement is modeled as the ground truth Euclidean distance between two UWB-radios plus noise drawn from the distribution $\mathcal{N}(\mu = 0.13, \sigma^2 = 0.2)$. This distribution is based on the aforementioned literature and experience gained with the Decawave DWM1000 module at RCS. The noise draws from the same (constant) distribution over all ranges which can be considered as a conservative approach. In the Maze25x25 environment the range is limited to $10\,\mathrm{m}$ and in the Maze75x75 environment the range is limited to $15\,\mathrm{m}$.

4.2.2. Odometry Noise

The Pioneer3DX, as many robots other robots, is equipped with a common differential drive that induces the motion. In [15] a probabilistic motion model based on the kinematics of a differential drive is introduced. Probabilistic motion models have generally been used to sample quasi-random motion of robots in a particle filter. These models can also be used to generate odometry increments with added noise according to a proper model. A noised odometry increment $\begin{bmatrix}\delta x' & \delta y' & \delta\theta'\end{bmatrix}^\mathsf{T}$ can be generated given a ground-truth odometry increment $\begin{bmatrix}\delta x & \delta y & \delta\theta\end{bmatrix}^\mathsf{T}$ and the model from [15]:

$$\begin{bmatrix}\delta x' \\ \delta y' \\ \delta\theta'\end{bmatrix} = \begin{bmatrix}\cos\hat{\delta}_{rot_1} & 0 & 0 \\ \sin\hat{\delta}_{rot_1} & 0 & 0 \\ 0 & 1 & 1\end{bmatrix} \begin{bmatrix}\hat{\delta}_{trans} \\ \hat{\delta}_{rot_1} \\ \hat{\delta}_{rot_2}\end{bmatrix} \tag{4.1}$$

distance m	**mean error** μ	**standard deviation** σ
1	0.0149 m	0.0134 m
2	0.0374 m	0.0116 m
5	0.055 m	0.0115 m
10	0.0809 m	0.012 m

Table 4.2: Ultra-wideband error distributions in a LOS configuration. The tables shows the different values for μ and σ for the ranges 1 m, 2 m, 5 m, 10 m. Results are from [88].

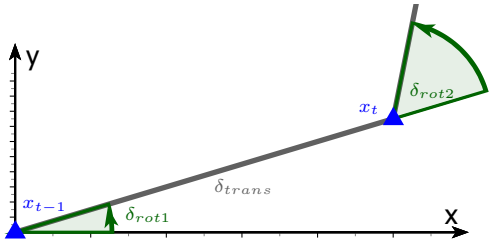


Figure 4.4: Thrun's probabilistic motion model for a differential drive.

in which:

$$\hat{\delta}_{trans} = \delta_{trans} + \epsilon_{trans} \qquad \epsilon_{trans} \sim \mathcal{N}\left(0, \sigma^2_{trans}\right) \tag{4.2}$$

$$\hat{\delta}_{rot_1} = \delta_{rot_1} + \epsilon_{rot_1} \qquad \epsilon_{rot_1} \sim \mathcal{N}\left(0, \sigma^2_{rot_1}\right) \tag{4.3}$$

$$\hat{\delta}_{rot_2} = \delta_{rot_2} + \epsilon_{rot_2} \qquad \epsilon_{rot_2} \sim \mathcal{N}\left(0, \sigma^2_{rot_2}\right) \tag{4.4}$$

and from [15] the following approximations are used:

$$\sigma_{rot_1} = \alpha_1 \left|\delta_{rot_1}\right| + \alpha_2 \delta_{trans} \tag{4.5}$$

$$\sigma_{rot_2} = \alpha_1 \left|\delta_{rot_2}\right| + \alpha_2 \delta_{trans} \tag{4.6}$$

$$\sigma_{trans} = \alpha_3 \delta_{trans} + \alpha_4 \left(\left|\delta_{rot_1}\right| + \left|\delta_{rot_2}\right|\right) \tag{4.7}$$

The meaning of the used variables can be deduced from figure 4.4. In all the Maze experiments the following noise parameters are used, derived from MRPT's* standard noise settings for this noise model:

$$\alpha_1 = 0.05$$
$$\alpha_2 = 0.0698132$$
$$\alpha_3 = 0.1$$
$$\alpha_4 = 0.00572958$$

4.3. Experiments

In this section the actual experiments are introduced. This consists out of visualizing the raw-odometry (or pose-trajectories) and introducing the parameters that were varied for the experiments.

For all four datasets, the SLAM algorithm will be ran and the quality will be assessed as discussed in section 4.4. For the maze experiments noise is added to the measurements as described in section 4.2.2 and 4.2. In the maze25x25 experiment the range of the UWB-radios is set to 10 m and in the maze75x75 experiment the range is set to 15 m. Each experiment is repeated until eleven successful runs were performed unless otherwise noted.

Influence of Range-measurement-noise

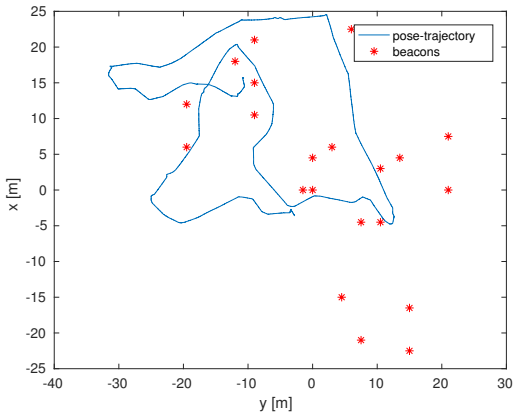
In the maze25x25 datasets the UWB-noise-parameter is adjusted. This is done to measure the influences on of the range-measurement-noise on the mapping performance. Each experiment with a new

*Mobile Robot Programming Toolkit is a C++ robotics library, which also implements [15]'s probabilistic motion model.

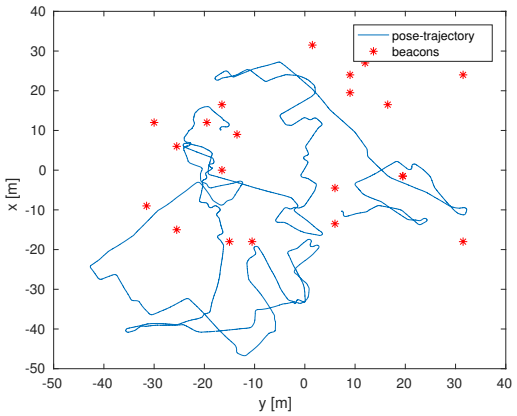
UWB-noise-configuration is repeated until eleven successful runs are performed. In addition to the standard parameters used in the first experiments, two extra UWB-noise-configuration will be used. These configurations are shown in 5.3.

4.3.1. Raw Pose-Trajectories

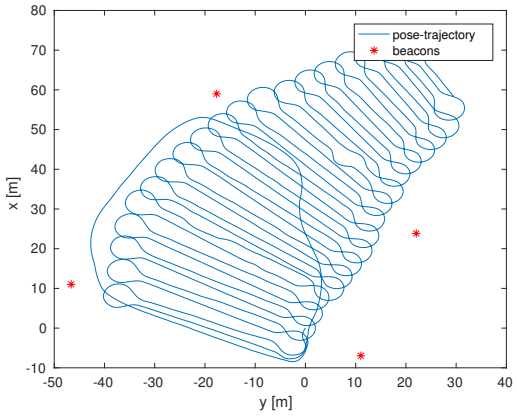
In the case of the Maze-experiments the noise varies per experiment generating and a new raw pose-trajectory is created each simulation run. Two examples of such trajectories are shown in figures 4.5a and 4.5b. In the Plaza datasets the raw pose-trajectory is exactly as shown in figures 4.5c and 4.5d.



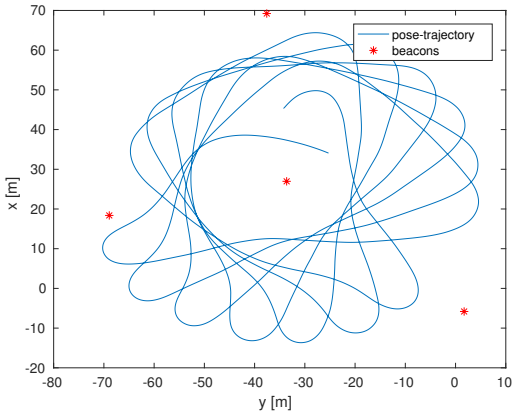
(a) An example of the raw uncorrected pose-trajectory in the Maze25x25 experiment.



(b) An example of the raw uncorrected pose-trajectory in the Maze75x75 experiment.



(c) Raw uncorrected pose-trajectory in the Plaza 1 experiment.



(d) Raw uncorrected pose-trajectory in the Plaza 2 experiment

Figure 4.5: The raw pose-trajectories for all four experiments. In the two Maze-experiments the odometry noise is generated through the introduced noise models. Note that the red asterisks denotes the locations of the UWB-radios in the global ground-truth frame, these locations are initially unknown and only displayed here as a visual reference. The pose-trajectory is plotted in the frame of the first odometry measurement.

4.4. Assessing Range-SLAM Performance

For all four datasets a ground-truth is available. The ground truth set is used to assess the performance of the SLAM algorithm. Further more the pose-trajectories can be visually compared to the ground-truth trajectory. The assessment on building grid-maps is only done by visual inspection of the produced

maps.

4.4.1. A Metric for Evaluating Range-SLAM

The most straightforward manner to assess the quality of a SLAM-graph is to compute the Root-Mean-Square Error (RMSE) of each variable in the graph with respect to its counter-part in the ground-truth set. Formally this equals to calculating the error of each pose and beacon-location in the same reference frame. The problem with this performance-metric is that it generates a non-objective error as a single error in an early pose is added to each of the consecutive poses. If the same error would occur in one of the later poses, the RMSE would be significantly lower, or at least different, compared to the first scenario. Secondly, one needs to be sure that it is straightforward to find a transform between the ground-truth frame and the global SLAM frame. This is not always the case. The performance-metric proposed in [44] suggests to calculate and a RMSE based on the relative displacements instead of the whole trajectory.

The second notion suggested in [44] is that often, the interest does not lie in to checking the quality of the pose-trajectory, but rather the other features, such as the locations of beacons. Also, commonly the error is based on the pose-trajectory but a scalar value that represents both the error in translation and rotation is hard to be interpreted. Often the error calculated as a weighted sum of both errors:

$$\epsilon(\mathbf{x}) = \frac{1}{T}\left(\sum_{t=0}^{T-1} trans\left(\hat{\mathbf{x}}_{t,t+1} \ominus \mathbf{x}_{t,t+1}\right) + \sum_{t=0}^{T-1} rot\left(\hat{\mathbf{x}}_{t,t+1} \ominus \mathbf{x}_{t,t+1}\right)\right) \tag{4.8}$$

In this equation the $trans(\cdot)$ and $rot(\cdot)$ weigh the error, $\hat{\mathbf{x}}$ is the SLAM-estimate of the pose, $\mathbf{x}$ is the ground-truth and the $\ominus$ operator is inverse of the motion composition operator $\oplus$. The $\ominus$ operator is needed to properly calculate the difference between two poses as it is a non-euclidean measure. In this error definitions there still remains a dubious addition of a euclidean distance error (distance between two points) to a angle-difference (difference between two headings). It is like adding an apple to a pear.

In the general case of range-based SLAM, however, the interest actually lies on how well the beacon-locations λ are estimated:

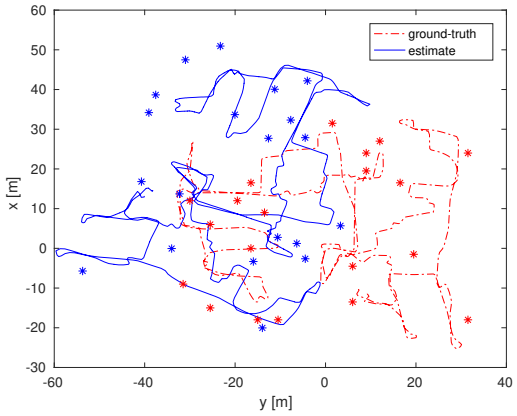
$$\epsilon(\lambda) = \sqrt{\frac{1}{k}\sum_{j=1}^{k}\left(\hat{\lambda}_j \ominus \lambda_j\right)^2} \tag{4.9}$$

In this equation the $\ominus$-operator simply calculates the euclidean distances between the points that represent the beacon-locations λ. Also, since we only consider points, the error is measured as simply a distance, just apples.

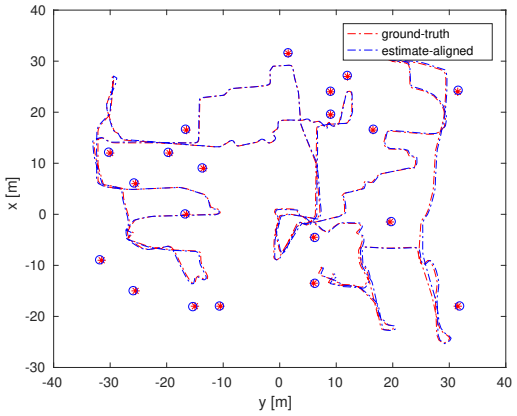
Following the train of thought on calculating relative displacement between poses, we introduce a new metric for the RMSE of beacon-locations. Our performance-metric is based on the relative distances between beacon-locations and therefore the calculation does not need a shared reference frame. For each beacon, in its own frame, the euclidean distance to all other beacons is calculated and compared to the ground truth:

$$\epsilon(\lambda) = \sqrt{\frac{1}{k^2-k}\sum_{j=1}^{k}\sum_{(j=1)\setminus\{i\}}^{k-1}\left((\hat{\lambda}_i \ominus \hat{\lambda}_j) - (\lambda_i \ominus \lambda_j)\right)^2} \tag{4.10}$$

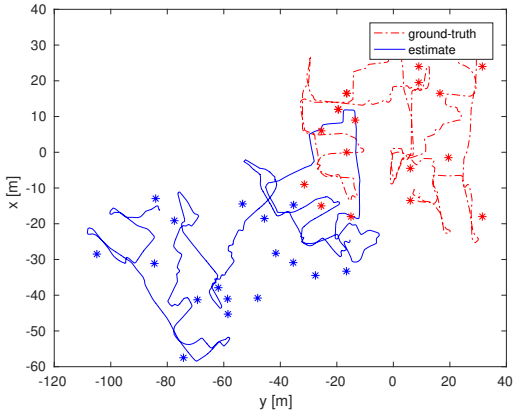
This error-metric is independent of a reference-frame and simplifies error calculation by omitting the need for a finding a transform between the global estimated frame and the global ground-truth frame. For all experiments, both the RMSE of the path-translation (note that the rotational error is ignored as done often [44] to prevent adding pears to apples) in the global frame *and* the RMSE of the beacon-locations based on relative distances. The RMSE of the pose-trajectory is calculated in the global frame so it can be compared to other researches, especially for the Plaza datasets.



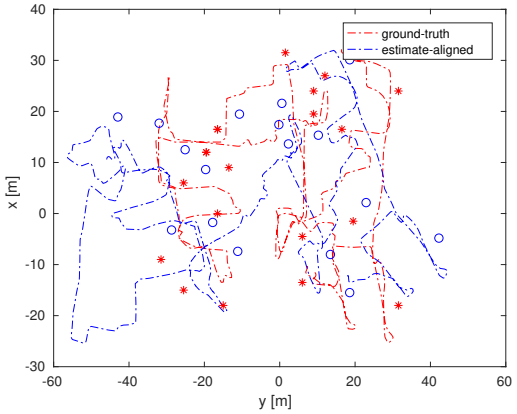
(a) Beacons successfully located, $\epsilon(\lambda) = 0.304$



(b) 4.6a aligned with ground-truth, $\epsilon(\lambda) = 0.304$



(c) Beacons unsuccessfully located, $\epsilon(\lambda) = 11.860$



(d) 4.6c aligned with ground-truth, $\epsilon(\lambda) = 11.860$

Figure 4.6: Visualization for two different $\epsilon(\lambda)$ values using the proposed performance-metric. Note that figures 4.6a and 4.6b display the same data, as do figures 4.6c and 4.6d. As a visual aid, the locations of the beacons in the SLAM frame are aligned with the beacons in the ground-truth frame. Because the data-association is known, as each beacon has an unique ID, a 2D rigid transformation can be found between the two sets of beacon locations that make the beacons overlap. If the beacon locations are faulty, finding the transform between the beacon-location-set and the ground-truth will result in a bad overlap, this is depicted in figure 4.6d.

4.4.2. Visually Assessing Performance

For illustrative purposes, the ground-truth and pose-trajectory are aligned to visually demonstrate the properness of the solution. An example of this alignment is shown in figure 4.6 and is based on finding the transformation between two point-clouds with known data-association. LiDAR observations, which are only available in the Maze-experiments, will also be plotted along the pose-trajectory to demonstrate the properness of the pose-trajectory and the resulting grid-map.

4.4.3. Summary

In this chapter the experiments are introduced. In total there are four environments: Maze25x25 and Maze75x75 which are created in a simulation environment, the other two environments are Plaza 1 and Plaza 2, which were created using an autonomous mower at Carnegie Mellon University. The two latter have no LiDAR-data and therefor the quality is only assessed through the RMSE of the pose-trajectory in a global-frame *and* the relative RMSE of the beacon-locations. For the Maze experiments the generated grid-maps are also plotted in addition to the previously mentioned RMSE-metrics. The

range-noise parameter is also varied to study the effects of an increasing error on range-measurements. In appendix C the work done towards setting up a new, real, experiment is shown.

5

Results & Discussion

This chapter is divided over two parts. In the first part the results will be shown and analyzed. For the Maze experiments the focus lies on the performance and the influence of noise. The Plaza experiments are mainly done for validation purposes and performance comparison to other algorithms on the same datasets. At the end of the first part, the algorithm's capability to build grid-maps and sub-maps is also demonstrated. The second part of the chapter is a discussion on the results.

5.1. Tabulated Results

This section shows the results using the performance benchmarks introduced in section 4.4. Eleven consecutive simulations were done in the Maze25x25 environment and all of them were successful. For the Maze75x75 twelve experiments were done of which one resulted in a bad mapping, this result is shown in figure 4.6c and is left out the of the tabulated data as it can be considered an outlier. For both Plaza experiments one run was done as there is only one dataset. The results are shown in table 5.1. In table 5.2 the results of the proposed algorithm is compared to the results of different algorithms that are also benchmarked on the Plaza datasets.

dataset	**trajectory length**	**error** $e(\lambda)$	**error** $e(\mathbf{x})$
Maze25x25	194 m	(0.1942 ± 0.0189) m	(0.247 ± 0.279) m
Maze75x75	621 m	(0.2042 ± 0.0389) m	(0.393 ± 0.381) m
Plaza 1	1860 m	4.6886 m	1.1260 m
Plaza 2	1354 m	4.9154 m	0.7106 m

Table 5.1: Results using HRL-SLAM algorithm and standard noise-parameters.

SLAM	**Plaza 1 error** $e(\mathbf{x})$	**Plaza 2 error** $e(\mathbf{x})$
Dead Reckoning (raw odometry)	15.92 m	27.28 m
HRL-SLAM	1.1260 m	0.71 m
Batch optimization [69]*	0.79 m	0.33 m
Spectral Learning [70]*	0.79 m	0.35 m
MetropolisHastings-SLAM [71]*	(0.33 ± 0.04) m	(0.36 ± 0.03) m

Table 5.2: Plaza 1 and 2 results compared to other SLAM-algorithms. * denotes that the range-measurements were pre-processed and/or interpolated measurements were added. The results of the algorithms on the bare data is unknown.

General performance

From table 5.1 it can be seen that the HRL-SLAM algorithm can locate beacons and in turn correct the pose-trajectory up to a certain accuracy. This is demonstrated by the low values of $e(\lambda)$ and $e(\mathbf{x})$. It is interesting to see that even though $e(\lambda)$ in the Plaza datasets is relatively high, indicating errors in the beacon-location, the pose-error $e(\mathbf{x})$ is significantly smaller. This is because the beacons are well placed and the positive bias in the measurements can be partly compensate for with the range-error from other measurements. This compensating effect is most noticeable the Plaza 2 experiment, as the central beacon-location is estimated accurately while while the "outer" beacons show the positive bias in location. This is illustrated in figure 5.2d. In [74] examples are shown of beacon locations that were also estimated using non-linear optimization (very comparable to HRL-SLAM) or direct trilateration. The compare section in that paper also shows relatively large errors of the beacons-locations compared to the ground truth, although not explicitly calculated, in combination with low pose-trajectory errors. The beacon-location error in the Plaza datasets can be attributed to the positive bias in the range measurements that can be up to several meters [12]. The fact that in [69, 70, 71] preprocessing of the data or data-interpolation was used suggests that their better results were achieved because of those measures. These differences are tabulated in table 5.2.

Performance with varying range-measurement noise

In table 5.3 it can be seen that the error increases if the quality of the range-measurement is decreased. Once the noise level of $\mathcal{N}(0.53, 0.8)$ is reached the beacon-estimation becomes very unreliable or fails all together, making the system fail. Such noise-levels are very uncommon using UWB-range measurements. The positive bias observed in the Plaza datasets has been tried to mimic using a Gaussian noise with a increasing non-zero mean. In table 5.3 it can be observed that an increase of the bias also increases the the mean of the error. The effect on the beacon-locations is visualized in figure 5.1.

noise $\mathcal{N}(\mu, \sigma^2)$	**error** $e(\lambda)$	**error** $e(\mathbf{x})$	**beacons**	**success-ratio**
$\mathcal{N}(0.13, 0.2)$	(0.1942 ± 0.0189) m	(0.247 ± 0.279) m	(18 ± 1)	100 %
$\mathcal{N}(0.27, 0.4)$	(0.4083 ± 0.0672) m	(0.416 ± 0.477) m	(15 ± 2)	91.67 %
$\mathcal{N}(0.53, 0.8)^*$	(1.4909 ± 0.5721) m	(1.6578 ± 0.8985) m	(9 ± 2)	36.36 %

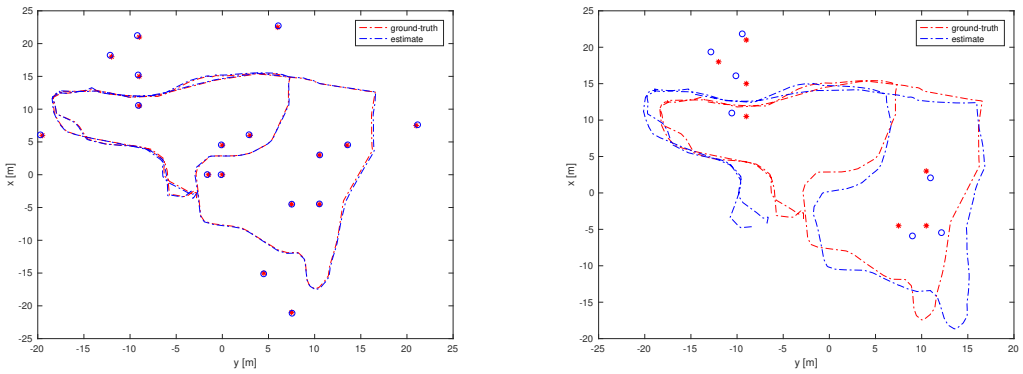
Table 5.3: Result of varying UWB noise-configurations in the Maze25x25 environment. * indicates that the results are based on 4 successful runs out of a total of eleven runs. Further experimentation at the noise level $\mathcal{N}(0.53, 0.8)$ was discontinued as it was clear the performance degraded beyond a usable level.

5.2. Visualized Results

The results are also visualized using the overlapping strategy explained in section 4.4. The resulting overlaps can be seen in figure 5.2. If the resulting pose-trajectories are compared to those in figure 4.5 the improvement is visually clearly observable. In figure 5.1b we see that beacon-locations share a similar constant off-set as can be observed in the Plaza experiments. The discrete jumps and in pose-trajectories such as visualized in figure 5.2c are artifacts because HRL-SLAM only records poses at instances at which a range-measurement was received. The Plaza datasets, especially Plaza 1, contains parts where there were no range measurements received for a while. This also occurs in the maze-experiments but it is shown less obviously. In figure 4.5b at the coordinates $(30, -25)$ a piece of trajectory can be seen that is not shown in figure 5.2b.

In the Maze experiments a short-range LiDAR was also used to record data. The result of plotting the LiDAR observation along the optimized trajectory path can be seen in figure 5.3d and 5.3f. The visualizations were made using RVIZ, the default visualization tool in the ROS-environment.

In figure 5.4 the change of covariances are observed and the pose-trajectory correction is visualized after a loop has been closed.



(a) $\mathcal{N}(0.13, 0.2)$ **(b)** $\mathcal{N}(0.53, 0.8)$

Figure 5.1: On the left there is the default parameter set $\mathcal{N}(0.13, 0.2)$, on the right one of few successful attempts with $\mathcal{N}(0.53, 0.8)$. Red depicts the ground-truth and blue the estimates of both the pose-trajectory as well as the beacon-locations.

5.2.1. Generation of Sub-maps

This section is used to demonstrate HRL-SLAM's ability to generate sub-maps given an observed beacon λ_j. Three resulting sub-maps are depicted in figure 5.5 given beacon observations λ_5, λ_{13} and λ_8 in an instance of the Maze25x25 experiment.

5.3. Discussion

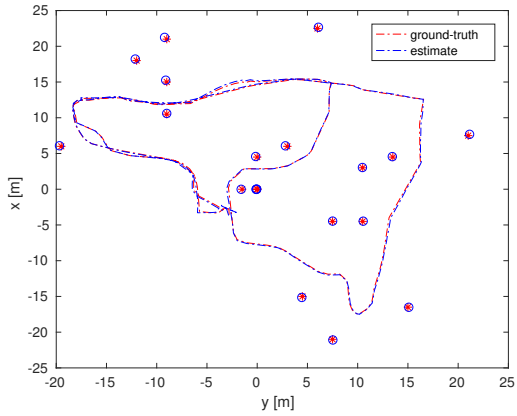
Many algorithms within robotics assume the availability of an environment model. In most cases this is in the form of a 2D map such as an occupancy grid-map. Creating metrically consistent grid-maps of indoor environments have proven to be a challenge using a LiDAR in combination with an ego-motion sensor, especially when low-grade sensors are used. In this thesis a Hybrid-Range-LiDAR SLAM algorithm was designed to determine the possibilities of enhancing the performance of a common grid-mapping setup by the addition of an UWB-range-sensor. Adding a new sensor did not only lead to building better grid-maps, but it also opened up ways to develop interfaces to partition spatial models into smaller spatial models - enhancing the scalability of both localization and mapping systems.

In the remaining part of this chapter the following subjects are discussed;

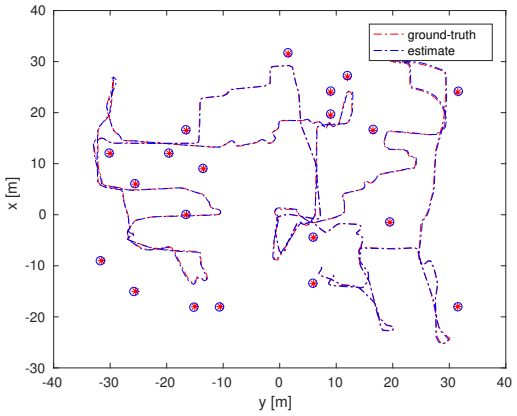
- General mapping performance using HRL-SLAM.
- On-line beacon-localization and the spectral filter.
- Improvements to scalability of SLAM and global robot-localization.
- Validity of simulation and Plaza experiments.
- Results of simulation and Plaza experiments.
- The benchmarks metric $e(\lambda)$.

5.3.1. Mapping Performance using Ultra-wideband Radios

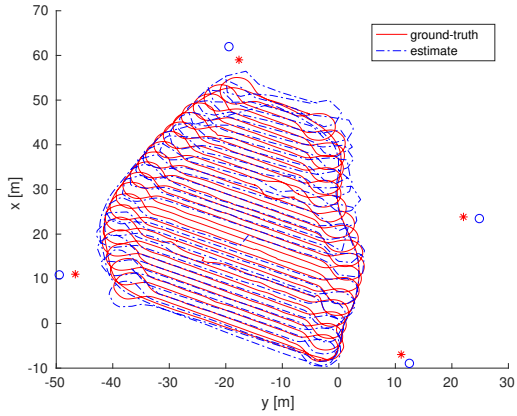
The experiment demonstrated that range-measurements from UWB-radios can be used to estimate beacon-locations *and* correct the pose-trajectory. In the case of the Maze experiments, it is shown that the range-corrected pose-trajectory is accurate enough to directly plot the LiDAR observations and still retrieve a visually proper grid-map of the environment. In many other implementations the creation of a proper grid-map requires an extra computation depending on whether the scans were already aligned (for pose-correction).



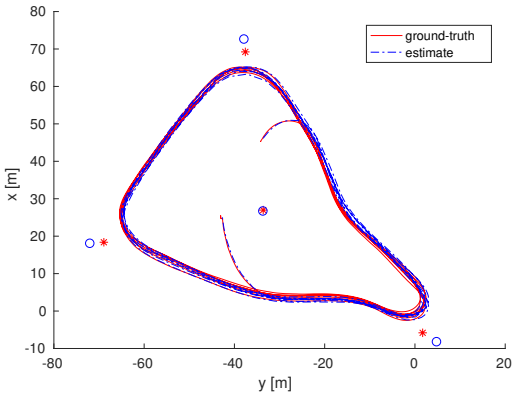
(a) $\epsilon(\lambda) = 0.1964$, $\epsilon(\mathbf{x}) = 0.0184$



(b) $\epsilon(\lambda) = 0.2067$, $\epsilon(\mathbf{x}) = 0.0401$

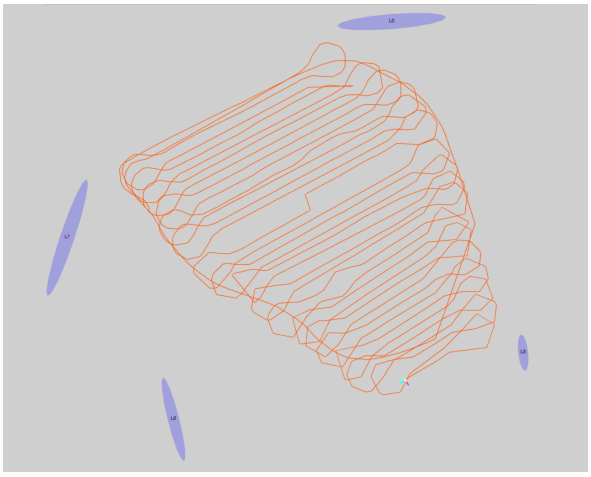


(c) $\epsilon(\lambda) = 4.6886$, $\epsilon(\mathbf{x}) = 1.1260$

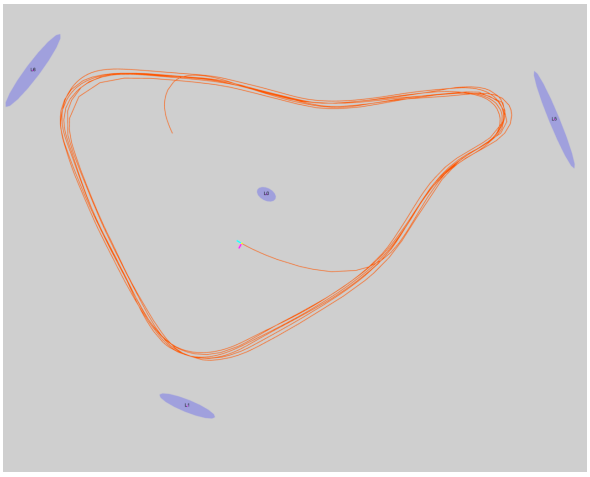


(d) $\epsilon(\lambda) = 4.9154$, $\epsilon(\mathbf{x}) = 0.71$

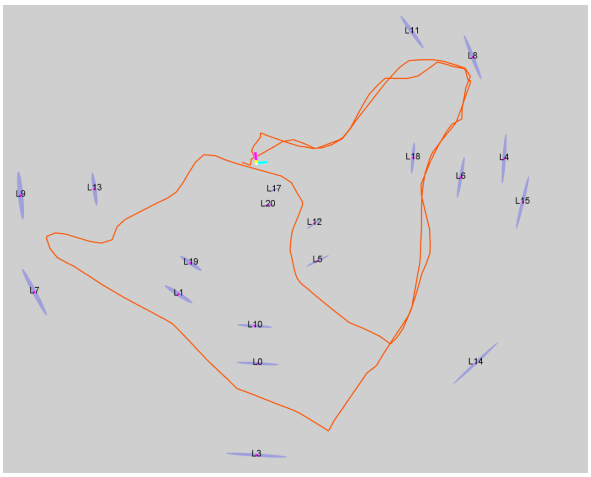
Figure 5.2: The top row shows two instances of the two maze experiments and the error $\epsilon(\lambda)$. Because the error is small and the relative distances between the beacons are close to the ground truth a neatly fitting overlap is found. In the case of the Plaza experiments $\epsilon(\lambda)$ is rather large and therefore it is not possible to neatly overlap the pose-trajectories and beacon -locations.



(a) Plaza 1.



(b) Plaza 2.



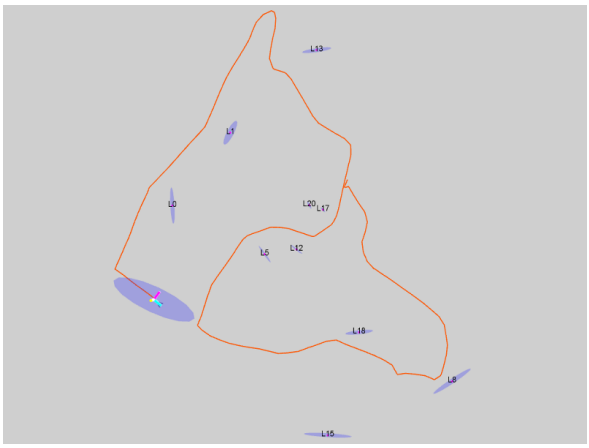
(c) Maze 25x25 pose-trajectory.

(d) Maze 25x25 grid map.

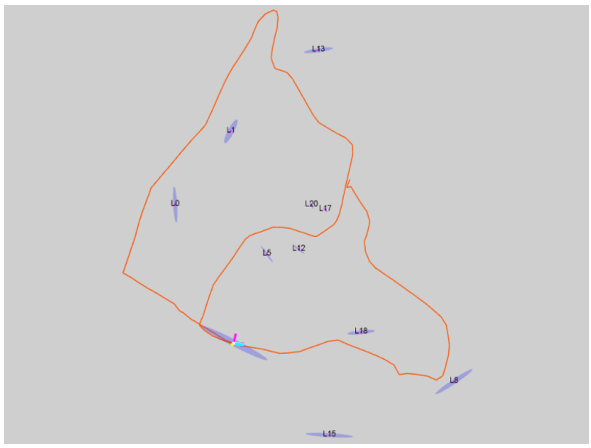
(e) Maze 75x75 pose-trajectory.

(f) Maze 75x75 grid map.

Figure 5.3: The trajectory visualized in RVIZ. The blue ellipses show the one standard deviation uncertainty around the estimated beacon-location. The uncertainties of all poses is omitted for clarity of the illustration. The grid maps were constructed by plotting the LiDAR observations as described in section 3.3.



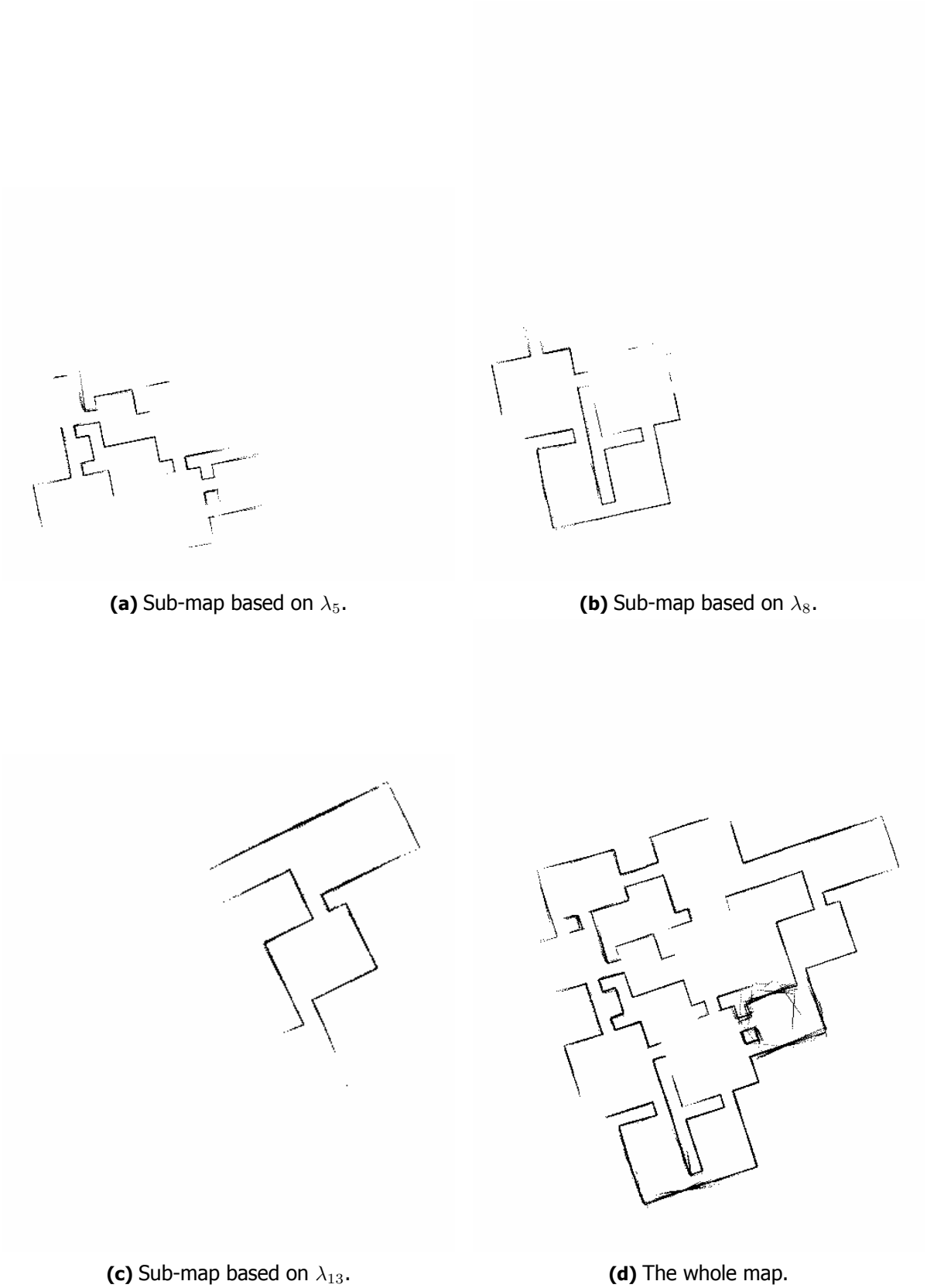
(a) Pose covariance before closing the loop.



(b) Pose covariance after closing the loop, e.g. re-encountering a located beacon.

(c) Visualization of the correct pose-trajectory and influence on beacon locations λ_0 and λ_1

Figure 5.4: Figures 5.4a and 5.4b show how the uncertainties (the covariance elipses) change after the algorithm implicitly closes a loop by re-encountering a beacon it already located before. As expected, the covariance shrinks and the pose trajectory is corrected such that the error in the non-linear optimization is minimized. In figure 5.4c the change of pose-trajectory and slight beacon-location changes are visible. These covariances are calculated with a prior constraint on the first pose, and so fort the marginals relevant to that pose (near λ_{20} in this case).



(a) Sub-map based on λ_5.

(b) Sub-map based on λ_8.

(c) Sub-map based on λ_{13}.

(d) The whole map.

Figure 5.5: Figures 5.5a, 5.5b and 5.5c demonstrate the sub-maps that can be generated given a range-measurement. Figure 5.5d shows the full map of the Maze25x25 experiment. The beacon-locations for λ_5, λ_8 and λ_{13} can be derived from figure 5.3d

Another important result is that all experiments have also been ran at over five times normal speed. The implementation does not employ any special efficient programming techniques such a GPU-computing. Much of the computational efficiency is expected to be a direct result from iSAM2's efficient column ordering* and fluid linearization of the Jacobians. But it should also be noted that because HRL-SLAM does not use the LiDAR for pose-correction, tasks such as scan-matching are not needed on a constant basis. In general, trilateration-computations are also significantly cheaper compared to using vision. In the case of vision one first needs to find the features through expensive computationally expensive image-processing and then compute a transform. All these steps are omitted because in range-based SLAM the range-measurements can be directly put into a simple set of constraints that can be added to the optimization.

From the results it is clear that a range-based SLAM solution using factor graphs can, at least in these cases, create appropriate models of the environment. Range-based systems can even be considered preferable if it is difficult to create reliable LiDAR-based loop-closures. Implicitly closing the loop with UWB-range-measurements has more certainty and is computationally simple opposed to a scan-matching procedure. The downside is however, that the system's performance becomes very dependable on the initial estimates for the beacon-locations λ. A wrongly initialized beacon can harm the optimization in an unrecoverable way similar to that of incorrect LiDAR loop-closures. However, it is more practical to do a manual correction on beacon locations opposed to removing loop-closure factors from a graph.

5.3.2. Estimating Beacon Locations using a Spectral Filter

Correctly initializing beacons is of the essence in HRL-SLAM. Where as most of the computational effort in LiDAR-based SLAM lies in scan-matching, beacon-localization is the resource-hog in HRL-SLAM. The fact that in the experiments we were able to create consistent maps using high-noise levels that are in reality uncommon for UWB, demonstrated that beacon localization was a success. The performance of the spectral beacon estimator initially proposed in [67] does not decrease unusably given the scalability adjustment and the completely different scenario. Visualizations of the spectral filtering, shown in figures 3.7a and 3.7c demonstrated that the spectral-filter *works* and filters out the interesting parts of the point cloud distribution.

The performance of the filter was also greatly improved by accessing the pose variables from the optimized set of poses opposed to relying on the dead-reckoning estimate. This was only possible because of the use of factor graphs opposed to the original Kalman-filter approach. This significantly improved the ability to estimate beacon-locations in general but especially when re-encountering an undiscovered beacon after a long loop.

During testing, it was also noticed that the spectral filter notably delayed finding a beacon if the pose-trajectory was collinear. This effect was further amplified by using the eigenvalues of the spectral filter's adjacency matrix as another decision aid, as described in [67]. This way the odds of improperly initializing a beacon, at the wrong side of the pose-trajectory, was significantly.

5.3.3. Improvements to Scalability of SLAM and Global Robot-Localization

Scalability is one of the open-problems in SLAM. Scalability can limit SLAM both in meeting real-time requirements but also because a maximum data storage limit can be exceeded. The input data is infinite as long as the robot operates. Where filter-based range-SLAM systems failed the on-line computability requirement due to growing covariance matrices (Kalman-filter-based SLAM) or very slow batch-optimization sequences (more specifically, [69] took over 9000 seconds in total to do the full optimization of the Plaza 1 scenario, which is beyond on-line computation), HRL-SLAM was able to optimize both datasets at least five times real-time. This is improvement over [69] is contributed to the implementation of iSAM2.

What HRL-SLAM does not take into consideration is any kind of tactic to remove information. In

*Column ordering happens to keep the information-matrix as much as possible in a upper-triangular form which makes the Cholesky decomposition significantly more efficient.

graph-SLAM graph-pruning is a method that removes variables from a graph in such a way that most information is kept. Most approaches that reduce the amount of variables in a factor graph are based on Chow–Liu trees [1]. HRL-SLAM should eventually also implement such a function to enhance the scalability. Important is that graph-pruning this does not necessarily imply *life-long mapping*. The road to life-long-mapping requires an answer to a more fundamental question: When does SLAM stop?

In this thesis scalable localization is based on creating an appropriate small map given a beacon observation. The observation of a beacon, the graph-structure and a definable neighborhood function allow the retrieval of a subset of variables and related constraints needed to create a small accurate map. The implementation in HRL-SLAM only retrieves the values of the variables in the subset and neglects any further processing before plainly plotting it using the grid-mapper. The neighborhood function is simple as the subset consists of the pose-variables that are within a defined euclidean-distance of the beacon-variable. Since it is very unlikely that a robot observes a beacon without actually being in the beacon's perimeter, this is a safe way to generate a sub-map. MCL in a small map such as depicted in figures 5.5a, 5.5b and 5.5c is more efficient opposed to spreading a population of the whole map such as depicted in 5.5d. This should significantly improve the robustness and convergence speed in which a robot can recover from the kidnapped robot situation.

The downside of the current implementations is that LiDAR observations that fall outside the perimeter of any beacon are never found in a sub-map. Then again, this information could also be seen as an indication where the robot could be: the full-map minus all the space that is also covered by sub-maps.

5.3.4. Validity of the Maze and Plaza experiments

The Plaza experiments demonstrated that at least in two instances, the algorithm successfully estimated the positions of beacons and corrected its own pose-trajectory. As mentioned earlier however, it should be noted that it is not uncommon for SLAM to succeed and later fail a similar dataset. To gain statistical trust and a truly demonstrated robust SLAM algorithm, real experiments should be repeated rigorously. This is uncommon in SLAM-research. The result of HRL-SLAM should therefore be taken with a grain of salt in the case of these two instances.

In the case of simulation however, the HRL-SLAM has proven to be surprisingly robust. With success-rates for mapping over $90\,\%$ the algorithm performed very well in two differently sized environments. Change of the range or its noise-properties did not dramatically change these results up until noise-parameters exceeded to something that can be considered unrealistic. But the simulation experiments have their downsides as well. While the used motion-model is generally accepted and verified and creates realistic noised-odometry, the UWB-noise implementation is rather simple and incomplete.

To further validate the performance of HRL-SLAM the simulation should be extended to incorporate a better UWB-model. This should include multi-path effects, differentiation between NLOS and LOS signals, the range-measurements should be more infrequent (sometimes include a period of sensor silence, no measurements). A different, arguably better, approach is to set up a real experiment using real UWB-radios. This approach has been worked on and the work towards it is described in appendix C.

5.3.5. Results of the Maze and Plaza Experiments

From table 5.2 can be concluded that HRL-SLAM performed decent compared to other algorithms performing on the same dataset. However, it should be noted that the comparison is not exactly fair as HRL-SLAM uses the raw range-measurements directly opposed to preprocessing or interpolating for extra measurements such as the other algorithms did. The focus, however, was on verifying the performance of HRL-SLAM on a real dataset as a form of algorithm validation. Whether the performance difference is algorithmic of nature or just a case of smart-filtering is unclear. The reason that the pose-trajectory error $e(\mathbf{x})$ is significantly smaller than the beacon-error $e(\lambda)$ can be explained by the fact that the beacons are well placed and have long range. This way range-measurements can compensate each other's errors.

The parameters in the Plaza experiments received a slightly coarser grid-cell size in the voting

mechanism (0.5 m in the Maze experiments opposed to 1 m in the Plaza experiments) but did not require any further changes to work appropriately.

The Maze and Plaza experiments differ significantly in various ways. The pose-trajectories and beacon-locations are very different and also the characteristics of the range-sensor differ. Therefore there are no valuable conclusions that can be drawn from comparing the experiments.

5.3.6. Relative Distances Between Beacons as a Benchmarking Metric

Many SLAM benchmark-metrics are based on the pose-trajectory and more specifically just the translational component. This is mainly because the quality of a SLAM-algorithm is judged on how well it can build a grid-map. In HRL-SLAM there is not only a grid-map and a pose-trajectory that can be assessed, but also the locations of the beacons. This applies to any feature-based SLAM algorithm. In the case of range-based SLAM however, a beacon-map is far more information efficient opposed to a full grid-map when it comes to assessing the map quality. In HRL-SLAM, wrongly estimating the positions of the beacons relative to each other is also the worst thing that can happen as it is not only wrong, but will also very likely corrupt future pose-trajectories and range-based localization instances.

The Plaza experiments show, however, that even though the beacon-estimates are significantly off, the pose-trajectory errors are still reasonably small. This only illustrates that the pose-trajectory error should be taken with a grain of salt. In future localization cases, using the beacon positions such as found in the Plaza experiments, will always lead to uncertain localization as the beacon-locations simply are a few meters off. It is also expected that other trajectories in the Plaza experiment with the same beacon configuration could lead to different results in terms of beacon-location quality.

6

Conclusion & Future Work

6.1. Conclusion

In this thesis it was questioned how an UWB-range-sensor could contribute to the mapping and localization performance of robots in indoor, usually cluttered, environments. By designing a range-based SLAM algorithm we have demonstrated that an UWB-range-sensor can significantly improve the consistency, accuracy and computation-time of robotic mapping and localization. The structure of HRL-SLAM also allowed the implementation of a sub-mapping strategy and introduced handles to incorporate concept as hierarchical and semantic mapping.

The design of the SLAM-algorithm is based on a literature review that suggests, as common in SLAM, to model the problem as a factor graph and incrementally use non-linear optimization to retrieve the optimal configuration with the highest likelihood. The factor graph approach has three major advantages; firstly, the inference calculations could be done using the sum-product algorithm, secondly it opens a way to easily add more sensors. In the current version of HRL-SLAM, only odometry- and range-measurements are used to optimize the graph. However, adding a new sensors to improve the estimate is as straightforward as creating an error-function that can be used as a constraint. Which can in turn be added as a factor to the factor graph. The third advantage is that by modeling the environment as a graph, a clear topological model is build which can help to retrieve only parts of an environment and use only locally relevant observations and constraints.

Through the conducted experiments we have demonstrated that Olson's spectral beacon-localization algorithm for acoustic ranging, in combination with the proposed scalability improvement, can also accurately estimate beacon-locations on-line when using UWB-radios. These locations are in turn used by the optimization-part of the SLAM algorithm to correct the pose-trajectory. The implicit loop-closures, because of the identifiability of the UWB-beacons, have also demonstrated to be very effective to improve both the pose-trajectories *and* beacon locations. The overall result is satisfying as the pose-correction is accurate to an extent that LiDAR-observations can be plotted without any correction and still produce optically good grid-maps.

The identifiability of the UWB-beacons in combination with the factor graph allowed to us to create a simple look-up mechanism to create small sub-maps. This was motivated as a way to lower the burden of large-scale MCL. The relationships between factors and variables, from which the sub-maps are generated, have in essence become a form of non-metrical spatial representation of the environment that holds a tight connection to the actual factor graph that does contain the metric information.

Finally, this thesis also introduced a benchmark metric for range-based SLAM that puts the essence much more on the beacon-locations, opposed to pose-trajectories. In the opinion of the author, this is a significantly more interesting metric, as wrongly estimating beacon-locations have a far larger consequence opposed to a wrong pose-trajectory.

The overall conclusion is that for applications in which it is possible to enhance the environment by

placing a set of UWB-radios, these radios improve the robots ability to build consistent environment models where they otherwise could not have done that using solely a LiDAR. The HRL-SLAM algorithm takes advantage of the UWB-sensor, by *smoothening* all data opposed to filtering, resulting in a graph that can be used for grid-mapping and large-scale localization.

6.2. Future work

The section future work is divided over two parts. The first part concerns practical features that can be incorporated into HRL-SLAM. The second part concerns the extended validation of HRL-SLAM on real data.

6.2.1. Improving HRL-SLAM

Grid-Maps to Occupancy Grid-Maps

The current version of HRL-SLAM focuses on robust beacon-localization and graph-optimization and not processing the LiDAR-observations to an accurate grid-map. However, the current implementation is able to create a grid-map with accurate positions of walls and other obstacles but it does not model the free space. Free-space information is crucial to create an usable occupancy grid-map. The current implementation uses ETH's* grid-mapping library which makes it straightforward to find free-cells given the end-points of laser-beams. The new information however, should be fused probabilistically using one of the many available methods for grid-mapping. An interesting start could be to use Google's Cartographer approach described in [31] or one could decide to stay within the theme of factor graphs as done in [90]. There are many other options available as well and a small research should help to find the solution that fits the best within the computational constraints (on-line or not on-line mapping for instance).

Scan-matching

In the current implementation of HRL-SLAM the LiDAR is only used to generate grid-maps from the LiDAR point-clouds. It could be interesting to integrate a graph-based LiDAR approach, such as Google's Cartographer [31], with range-measurements factors, such as those in HRL-SLAM. The downside is of course the increased computability, as scan-matching then becomes a task again, on the upside however, the real-time constraint of SLAM is not always there and sometimes it might be beneficial to use both LiDAR and range-measurements to generate a superior map off-line and use it for future reference. The graph-topology could be extendend such as examplified in figure 6.1. This approach is actually more true to the decription Hybrid-Range-LiDAR SLAM compared to the version dubbed in this thesis.

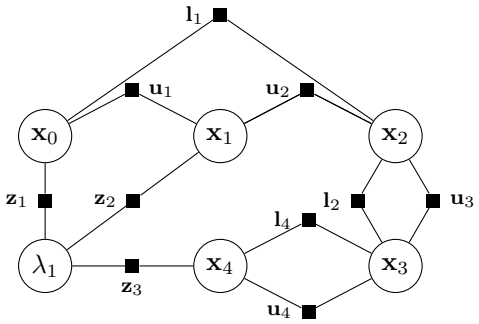


Figure 6.1: Example of the graph topology if odometry-, range- and LiDAR-factors are used. Scan-matching could also be used to improve the odometry-estimate.

*ETH Zurich, Swiss Federal Institute of Technology in Zurich

Semantic hierarchy

The graph-topology allowed us to use a simple version piece of information "I see λ_1" to create a small map of a local area. This is very similar to how humans do spatial understanding. Referring to its proper tag, such as *my house* or *my bedroom* triggers the brain to find the associated memories and allow us to construct a spatial model on demand. The way how sub-maps are generated in HRL-SLAM acts along the same mechanism. The sub-map-topology is defined by the neighborhood algorithm (could be a function, but it could also be a clustering or partitioning method) that defines what observations belong to which sets. There should be no constraints on creating disjoint sets or whatsoever, the resulting system should be a smart-lookup system that can decide which observation belong to each other and which spatial-constraints should be used to create a locally accurate map.

Using proper partitioning techniques, the metric factor graph should be segmented in smaller graphs that can be tagged with meta-data which makes it easy to look-up. If this meta-data can be made compatible to the descriptors humans use, a large leap forward can be made between mutual spatial understanding between humans and machines. In the opinion of the author, this is where HRL-SLAM should eventually work towards. This process can be accelerated by working towards the standard sets by institutions such as the Open Geospatial Consortium (OGC) which tries to set standards for how space should be partitioned given sensor observations. An example of such a partitioning is shown in figure 6.2.

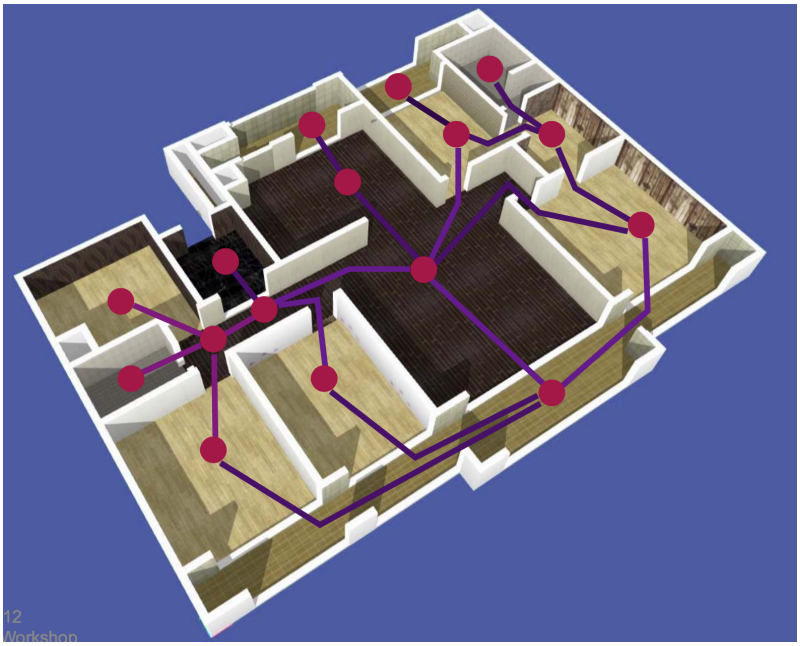


Figure 6.2: An example of indoor space segmentation using the standards set by Indoor GML. [91].

6.2.2. Extended Validation on Real Data

The difference on accuracy-results between the two Maze and Plaza environments is difficult to explain because experiments differ a lot. To improve the validation of the algorithm, an experiment using the actual Decawave DWM1000 modules in an indoor environment should be conducted. This will also check how well the algorithm can cope with NLOS situations and other UWB deficiencies. The consistency at which range-measurements occurred in the Maze experience are also unrealistic in real life. This was also observable by the radio-silences in the Plaza datasets. See appendix C for the work that has been done towards setting up a real experiment at Robot Care Systems bv.

A

Sparse optimization on manifolds

From a sparse SLAM-graph to a feasible optimization problem

As a refresher, optimization is the quest of finding an optimal state $\mathbf{x}^*$ such that a cost function $\mathbf{F}(\mathbf{x})$ is minimized:

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}}\, \mathbf{F}(\mathbf{x}) \tag{A.1}$$

Iterative solvers, such as the gradient descent methods mentioned before, are step-based solvers that try to find increments $\Delta\mathbf{x}$ such that $\lim_{n\to\infty}(\mathbf{x}_{n-1} + \Delta\mathbf{x}_n) = \mathbf{x}^*$. At each step n, $\mathbf{F}(\mathbf{x})$ is linearized around the estimated state $\hat{\mathbf{x}}$, then a $\Delta\mathbf{x}$ is proposed and is added to the current state estimate: $\hat{\mathbf{x}} = \hat{\mathbf{x}} + \Delta\mathbf{x}$. This process iterates until it converges or some other stop-criteria is met.

The determination of $\Delta\mathbf{x}$ and the approximation of $\mathbf{F}(\hat{\mathbf{x}} + \Delta\mathbf{x})$ vary per non-linear optimization algorithm, but all require the calculation of $\nabla\mathbf{F}$, and Hessian of $\mathbf{F}$, $\mathbf{H}_\mathbf{F}$:

$$\nabla\mathbf{F} = \left.\frac{\partial\mathbf{F}}{\partial\mathbf{x}}\right|_{\hat{\mathbf{x}}} \tag{A.2}$$

$$\mathbf{H}_\mathbf{F} = \left.\frac{\partial^2\mathbf{F}}{\partial\mathbf{x}^2}\right|_{\hat{\mathbf{x}}} \tag{A.3}$$

The *optimality* of $\Delta\mathbf{x}$ determines how well the optimization converges to a global extrema. The effect of a less optimal $\Delta\mathbf{x}$ can be softened by introducing accurate initial estimates *and* smaller steps of $\Delta\mathbf{x}$. However, both these additives are palliatives. All methods finding $\Delta\mathbf{x}$ do involve differentiating $\mathbf{F}(\mathbf{x})$ and equalling it to zero: requiring the inverse of $\nabla\mathbf{F}$ and $\mathbf{H}_\mathbf{F}$: a costly operation.

In many cases the cost function is expressed as the squared error $\mathbf{e}(\mathbf{x})$, and this is also often the case in SLAM:

$$\mathbf{F}(\mathbf{x}) = \frac{1}{2}\mathbf{e}(\mathbf{x})^\intercal\mathbf{\Omega}\mathbf{e}(\mathbf{x}) \tag{A.4}$$

in which Ω is a symmetric, positive-definitive matrix and is called the information matrix. Now $\nabla\mathbf{F}$ and $\mathbf{H}_\mathbf{F}$ can be determined using equation A.3:

$$\nabla\mathbf{F} = \left.\frac{\partial\mathbf{F}}{\partial\mathbf{x}}\right|_{\hat{\mathbf{x}}} = \left(\mathbf{e}^\intercal\mathbf{\Omega}\,\frac{\partial\mathbf{e}}{\partial\mathbf{x}}\right)\Bigg|_{\hat{\mathbf{x}}} = \hat{\mathbf{e}}\mathbf{\Omega}\mathbf{J} \tag{A.5}$$

$$\mathbf{H}_\mathbf{F} = \left.\frac{\partial^2\mathbf{F}}{\partial\mathbf{x}^2}\right|_{\hat{\mathbf{x}}} = \left(\frac{\partial\mathbf{e}}{\partial\mathbf{x}}^\intercal\mathbf{\Omega}\frac{\partial\mathbf{e}}{\partial\mathbf{x}} + \mathbf{e}^\intercal\mathbf{\Omega}\,\frac{\partial^2\mathbf{e}}{\partial\mathbf{x}^2}\right)\Bigg|_{\hat{\mathbf{x}}} = \mathbf{J}^\intercal\mathbf{\Omega}\mathbf{J} + \hat{\mathbf{e}}\mathbf{\Omega}\mathcal{H} \tag{A.6}$$

$\hat{\mathbf{e}}$ is the error with its Jacobian (first order derivatives) $\mathbf{J}$ and Hessian (second order derivatives) $\mathcal{H}$. Substituting these definitions into a second-order Taylor approximation of $\mathbf{F}(\hat{\mathbf{x}} + \Delta\mathbf{x})$ leads to an approximation of the cost.

On *sparsity*

The SLAM optimization problem can be represented by a cost function $\mathbf{F}(\mathbf{x})$ that is the sum $\mathbf{e}(\mathbf{x})_k$ for all K poses.

$$\mathbf{F}(\mathbf{x}) = \sum_{k=1}^{K} \mathbf{e}(\mathbf{x})_k^\intercal \mathbf{\Omega}_k \mathbf{e}(\mathbf{x})_k \tag{A.7}$$

This summed error is of the same form as equation A.4 since $\mathbf{e}(\mathbf{x})^\intercal$ is a vector of errors $[\mathbf{e}(\mathbf{x})_1, \ldots, \mathbf{e}(\mathbf{x})_K]$ and $\mathbf{\Omega}$ is a matrix with $\mathbf{\Omega}_1$ up to $\mathbf{\Omega}_K$ on the diagonal. Each error is a consists out of the actual error $\mathbf{e}(\mathbf{x})_k$ and its associated information matrix $\mathbf{\Omega}_k$ which can be *represented by a factor in a factor graph*.

The error function $\mathbf{e}(\mathbf{x})_k$ is in general a non-linear function and thus again a Taylor expansion is used to iterate $\mathbf{e}(\mathbf{x})_k$. Because of the Taylor expansion, $\mathbf{J}_{1:k}$ and $\mathcal{H}_{1:k}$ have to be determined. Due to the structure of the SLAM problem, variables are only locally constrained: pose $\mathbf{x}_i$ is, in a pose graph, only connected to $\mathbf{x}_{i-1}$ and $\mathbf{x}_{i+1}$: this limits the computation of the amount of derivatives significantly. Even if the observations of landmarks are included, the Jacobian will still be sparse, resulting in less computations and a sparse $\mathbf{H}_\mathbf{F}$.

On a *manifold*

Since the state is described in $SE(2)$, which consists of a translation vector, a rotation and a composition operator, optimization is less straightforward that it initially might seem. The problem lies partly in that translation and rotation do no commute. This means that it matters in which order the translation and rotation is done. Translating and then rotating is different from translating and then rotating. The second part of the problem actually applies mostly to the 3D case because of the over-parameterization of the problem using Euler-angles.

To overcome these problems, first a composition operator $\oplus$ is defined which defines how to *add* 2D poses T_1^w and T_2^w of the form $[x, y, \theta]$. Note that w refers to that the pose is defined as a translation and rotation with respect to the frame w. First the poses are described in $GL(3)$ (general linear group or order three) using 3×3 matrices, this gives:

$$T_1^w = \begin{bmatrix} R_1 & t_1 \\ 0 & 1 \end{bmatrix}$$
$$T_2^w = \begin{bmatrix} R_2 & t_2 \\ 0 & 1 \end{bmatrix}$$

In these equation t_1 is the translation and R_1 the 2D rotation matrix:

$$R(\theta_1) = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 \\ \sin\theta_1 & \cos\theta_1 \end{bmatrix}$$

We can define the transformation from 1 to 2 as:

$$\begin{aligned} T_2^1 &= (T_1^w)^{-1} T_2^w \\ &= \begin{bmatrix} R_1 & t_1 \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} R_2 & t_2 \\ 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} R_1^\intercal R_2 & R_1^\intercal (t_2 - t_1) \\ 0 & 1 \end{bmatrix} \end{aligned}$$

The second part concerns defining a derivative that can be used in the non-linear optimization part. Given the pose composition, we can also describe a pose composed with a very small pose (an infinitesimal small increment):

$$T(\delta) = \begin{bmatrix} \cos\omega\delta & -\sin\omega\delta & v_x\delta \\ \sin\omega\delta & \cos\omega\delta & v_y\delta \\ 0 & 0 & 1 \end{bmatrix} \approx \begin{bmatrix} 1 & -\omega\delta & v_x\delta \\ \omega\delta & 1 & v_y\delta \\ 0 & 0 & 1 \end{bmatrix}$$

If now we define a *2D-twist* vector $\xi = (v, \omega)$ and the matrix:

$$\xi \triangleq \begin{bmatrix} 0 & -\omega & v_x \\ \omega & 0 & v_y \\ 0 & 0 & 0 \end{bmatrix}$$

Now if we have a relatively big step t, we can split t into n smaller time steps that approximate to ξ.

$$T(t) = \lim_{n \to z} \left(I + \frac{t}{n} \xi \right)^n$$

If z would go to infinity the solution would become very accurate. The form of this series is actually the exponential function for the reals. The exponential function can also be defined for square matrices:

$$T(t) = \exp(t\xi) \triangleq \lim_{n \to \infty} \left(I + \frac{t}{n} \xi \right)^n = \sum_{k=0}^{\infty} \frac{t^k}{k!} \xi^k$$

This allows the mapping between 2D twist-vectors and 2D rigid transformations. Since we can now map between these two spaces, a cost function that is linear in the update δ can be used using the 2D twist parameterization.

Through the simplifications $R_1^\intercal(\theta) = R(-\theta)$ and $R_1^\intercal(\theta) R_2(\theta) = R(\theta_2 - \theta_1)$ the following *measurement estimator function* can be created:

$$\widehat{\Delta\xi} = h(\xi_1, \xi_2) = \begin{bmatrix} R(\theta_1)(t_2 - t_1) \\ \theta_2 - \theta_1 \end{bmatrix}$$

The set of poses $X = \{\xi_j\}_{j=1}^n$ that satisfy the constrains $\Delta\xi_i$ the best is the set of poses with the maximum likelihood. This set of poses can be found by optimization:

$$\mathbf{F}(X) = \frac{1}{2} \sum_i \| h(\xi_{j1}, \xi_{j2}) - \Delta\xi_i \|^2$$

Note that here a general least squares error function is used, but this can also be extended to other more tropical error functions such as a m-estimators.

One of the challenges lies in the fact that the measurement estimation function h is non-linear. As done often, the non-linear function can be minimized by using a generalized Taylor expansion:

$$h(\xi_1 \oplus \delta_1, \xi_2 \oplus \delta_2) = h(\xi_1, \xi_2) \oplus \{H_1 \delta_1 + H_2 \delta_2\}$$

In this equation $\delta_1, \delta_2 \in \mathbb{R}^3$ are *pose updates* and H_1 and H_2 are 3×3 Jacobian matrices. Further we define an update $\xi \oplus \delta$ in the coordinate frame of ξ, meaning:

$$\xi \oplus \delta = \begin{bmatrix} t + R(\theta)\delta t \\ \theta + \delta\theta \end{bmatrix}$$

The introduction of the $\oplus$ operator is needed because the poses $\xi \in SE(2)$ in which vector addition is not defined. Using $\oplus$ makes the math properly behaved and gives us a tool to add increments to the pose. The biggest challenge lies in finding the Jacobians H_1 and H_2. In this specific example the Jacobians can be calculated as explained in [24], but the derivation is omitted for clearness:

$$H_1 = \begin{bmatrix} R_2^\intercal R_1 & R(-\pi/2) R_2^\intercal (t_1 - t_2) \\ 0 & 1 \end{bmatrix}$$

$$H_2 = \begin{bmatrix} I & 0 \\ 0 & 1 \end{bmatrix}$$

The approximated objective function is now linear in the update δ, so we can rewrite the objective function as:

$$\begin{aligned}\mathbf{F}(\delta) &= \frac{1}{2}\sum_i \|h(\xi_{j1},\xi_{j2}) + H_{j1}\delta_{j1} + H_{j2}\delta_{j2} - \Delta\xi_i\|^2 \\ &= \frac{1}{2}\sum_i \|A_i\delta - b_i\|^2 \\ &= \frac{1}{2}\|A\delta - b\|^2\end{aligned}$$

The A matrix is $3m \times 3n$ and is composed of block rows:

$$A_i = \begin{bmatrix}\dots & H_{j1} & \dots & H_{j2} & \dots\end{bmatrix}$$

and

$$b_i = h(\xi_{j1},\xi_{j2}) - \Delta\xi_i$$

which are the predicted errors which are 3×1 in size. From here we can simply optimize the update δ by setting the derivative of $\frac{1}{2}\|A\delta - b\|^2 = 0$:

$$\begin{aligned}A^\intercal(A\delta^* - b) &= 0 \\ (A^\intercal A)\delta^* &= A^\intercal b \\ \delta^* &= (A^\intercal A)^{-1}A^\intercal b\end{aligned}$$

This is a linear system that can be solved, after that the estimate can be update by doing

$$\xi_i^{t+1} = \xi_i^t \oplus \delta_i$$

for all the poses ξ_i where t are the indexes of the iterations. The process of linearizion and optimization can be repeated until a certain threshold for the update size δ_i is reached. A quick sanity check reveals that h actually is linear in the rotations as $h_\theta(\xi_1,\xi_2) = \theta_2 - \theta_1$. The trick is to first linearly solve for the rotations and secondly solve the second linear system that is associated with the translations:

$$h(\xi_1,\xi_2) = R(-\theta_1)(t_2 - t_1)$$

In 3D, this does not hold. The translation will still be linear once the rotations are known, but the rotation part is non-linear. Luckily, sensors can give us reasonably accurate initial conditions which can be used to do for a non-linear approximation of the rotations.

B

Implementation of HRL-SLAM

The purpose of this appendix is to describe the workings of the HRL-SLAM algorithm using pseudo-code and process-flow charts. The appednix ends with some practical notes on the actual implemented application and the adjustable parameters. The algorithm is event driven and reacts to incoming data. From an algorithmic point of view, there are three sources of data: *odometry*, *range-sensor*, LiDAR of which only the latter two induce algorithmic action. In practice there is an extra source of information, the parameter set Θ, which can influence the behavior of HRL-SLAM significantly.

B.1. Algorithmic Flow in Pseudo-Code

To break down the algorithmic into a logical flow a couple of simplifications and definitions have to be made. It is assumed is always an odometry-reading $\mathbf{x}_o$ on the interval $0 : t$ available. Range-measurements $\mathbf{z}_{r,j,t}$ from beacon λ_j and LiDAR-observations $\mathbf{z}_{l,t}$ arrive at time t in an asynchronous matter. In general the frequency of LiDAR observations is higher than that of the range-measurements. The pose-change is defined as $\delta\mathbf{x}_t = \mathbf{x}_t \ominus \mathbf{x}_{t_q}$. Once the norm of $\delta\mathbf{x}_t$ exceeds a threshold defined in Θ, calculations will be done. $\delta\mathbf{x}_t$ usually refers to the change compared to the previous time the event happened: if at $t = 3$ a range-measurement was received and at $t = 7$ the next range-measurement of the same beacon is received, $\delta\mathbf{x}_7 = \mathbf{x}_7 \ominus \mathbf{x}_3$. The most influential parameter is $\mathbf{x}_{thresh}$ it is used to decide whether to process a measurement or not. In general $\mathbf{x}_{thresh}$ differs per sensor: LiDAR or range-measurement, but it is the same in the pseudo code for simplicity.

There are three special objects $\mathcal{G}$, $\mathcal{B}$, and $\mathcal{F}$ that hold data and have functions that use the data or operate on it. In principal, all data, is stored in $\mathcal{F}$. $\mathcal{F}$ contains values for all variables $\mathbf{x}$, λ and it contains the factors ϕ. Factors are the functions that are depended on only a few variables, the dependencies of the factors create the topology of the (factor) graph. The objects $\mathcal{G}$ or $\mathcal{B}$ do not store values for the variables, instead they retrieve the values they need from $\mathcal{F}$. This way $\mathcal{G}$ or $\mathcal{B}$ always use the most recent and accurate estimates: $\mathbf{x}^*$ λ^*.

The pseudo-code is divided into 2 parts: an umbrella application in algorithm 1, the beacon-localization function $locateBeacon(\mathcal{F}, \mathcal{B})$ in algorithm 2. The algorithm for $\mathbf{G}$ is included in pseudo-code 1 as it is rather small and simple. The optimization function $optimize(\mathcal{F})$ in based on GTSAM implementation of iSAM2 [8]. iSAM2's optimization algorithm is preceded by a validation on the validity of the optimization. This means that the application checks if the optimization is underdetermined[†], and if so it removes the unconstrained variables to make the optimization determined.

The umbrella application holds all the functionalities and leaves room to be extendend through specific service requests. The pseudo-code is shown in algorithm 1 and the accompanying graphical flow is depicted in figure B.1, the specific handeling of a range-measurement is depicted in figure B.2.

[†]If a system of linear equations has more unknown variables than equations the system is underdetermined.

Algorithm 1: The umbrella application

Data: Parameters Θ, data streams $\mathbf{z}_r$, $\mathbf{z}_l$, $\mathbf{x}$ and service request interrupt $\mathbf{q}$.
Result: Optimized pose-trajectory $\mathbf{x}^*$, set of beacon locations λ^* grid-map $\mathbf{M}$.
begin
`// connect to data streams, initialize objects` $\mathcal{F}$, $\mathcal{B}$ `and` $\mathcal{G}$.
initialization
Loop *listen for data*
switch *data* **do**
case $(\mathbf{z}_{r,j,t} \wedge |\delta\mathbf{x}_t| > \mathbf{x}_{thresh})$ **do**
if λ_j *not located* **then**
add $\theta_t(\mathbf{x}_t, \mathbf{z}_{r,j,t}, \delta\mathbf{x}_t)$ to $\mathcal{B}_j$
if $locateBeacon(\mathcal{F}, \mathcal{B}_j) = true$ **then**
λ_j is located
destroy $\mathcal{B}_j$
end
end
add $\phi_{o,t,t-1}(\mathbf{x}_t, \mathbf{x}_{t-1}, \delta\mathbf{x}_{t,t-1})$ to $\mathcal{F}$ `// Add odometry factor`
add $\phi_{r,j,t}(\mathbf{z}_{r,j,t}, \mathbf{x}_t)$ to $\mathcal{F}$ `// Add range factor`
$\mathbf{x}_t^* = \mathbf{x}_{t-1}^* \oplus \delta\mathbf{x}_{t,t-1}$ `// Initial estimate for new variable` $\mathbf{x}_t$
add $\mathbf{x}_t^*$ to $\mathbf{x}^*$
`// Optimize using iSAM2 and Powell's Dogleg method.`
update $\mathbf{x}^*$: $optimize(\mathcal{F})$
end
case $(\mathbf{z}_{l,t} \wedge |\delta\mathbf{x}_t| > \mathbf{x}_{thresh})$ **do**
$\delta\mathbf{x}_{t,l} = \mathbf{x}_t \ominus \mathbf{x}_t^*$ `// Calculate transform to most recent pose in` $\mathcal{F}$
convert $\mathbf{z}_{l,t}$ to point-cloud $\mathcal{P}_t$
add $\omega(\mathbf{x}_t^*, \delta\mathbf{x}_{t,l}, \mathcal{P}_t)$ to $\mathcal{G}$
end
case $\mathbf{q}$ **do**
process custom service request, e.g.
$\mathbf{M} = getGridMap(\mathcal{G})$
$\Sigma_{\mathbf{x}_t} = getMarginal(\mathcal{F}, \mathbf{x}_t^*)$
end
end
EndLoop

Algorithm 2: The beacon localization function.

Data: Parameters Θ
Function `locateBeacon(`$\mathcal{F}$, $\mathcal{B}_j$`)`**:**
Get values for all $\mathbf{x}$ related to a random set of measurements $\mathbf{z}_{r,j,t}$ with beacon j from $\mathcal{F}$
create an array Ψ of tuples $\psi(\mathbf{x}_t, \mathbf{z}_{r,j,t})$
for *each* ψ_m *in* Ψ **do**
for *each* ψ_n *in* $\Psi \setminus \psi_m$ **do**
find set of intersections $\{p_1, p_2\}$ between the circles $(\psi_m.\mathbf{x}, \psi_m.\mathbf{z})$ and $(\psi_n.\mathbf{x}, \psi_n.\mathbf{z})$
update the adjecency matrix A:

$$A(m,n), A(n,m) = \begin{cases} \{1,1\} & (\psi_m.\mathbf{x}, \psi_m.\mathbf{z}), (\psi_n.\mathbf{x}, \psi_n.\mathbf{z}) \text{ intersect} \\ \{0,0\} & \text{otherwise} \end{cases}$$

find the largest eigenvalue λ and associated eigenvector v
create discrete representation u of v

$$u(i) = \begin{cases} 1 & v(i) > \theta \\ 0 & \text{otherwise} \end{cases}$$

find optimal θ: $\max_{\theta \in u} v \cdot u$
remove intersections p_o that are related to an outlier measurement ($u(i) = 0$)
plot remaining inliers on a discretized grid as explained in 3.2.2
find coordinates λ_1 of cell with most hits
find coordinates λ_2 of cell with most hits excluding λ_1
if $hits(\lambda_1)/hits(\lambda_2) > 2$ **then**
Beacon found at λ_1
else
Beacon not found
return

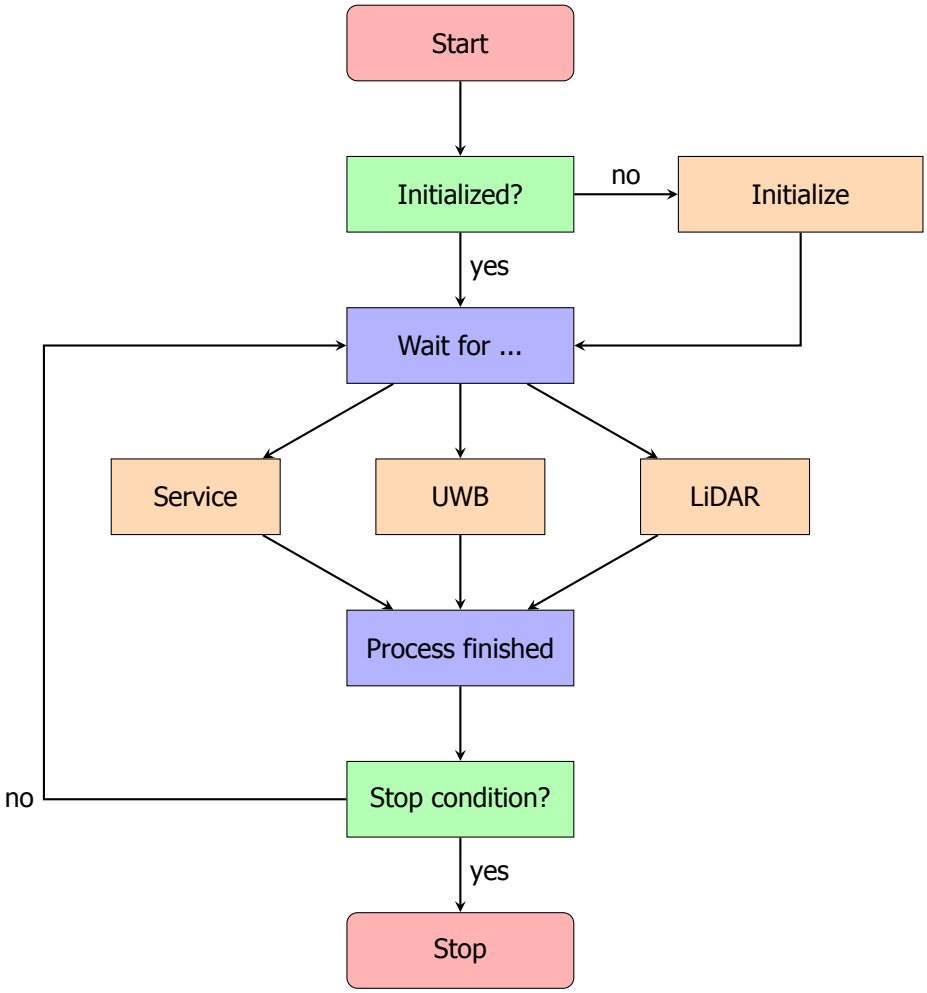


Figure B.1: Flow-diagram of the main application.

B.2. HRL-SLAM as a ROS-package

The actual implemented system is build on top of ROS [92] and GTSAM [24] using C++. The ROS-package is also depended on ETH's Grid Map Library [93] and the C++linear algebra library Eigen [94]. The system can be used as a standalone ROS-node given the required dependencies in table B.1 are available.

B.3. HRL-SLAM Parameters

The following table shows all the configurable parameters of HRL-SLAM. The default value, as used in all experiments unless noted otherwise, is also given. The parameters are configurable through a ROS-launch file.

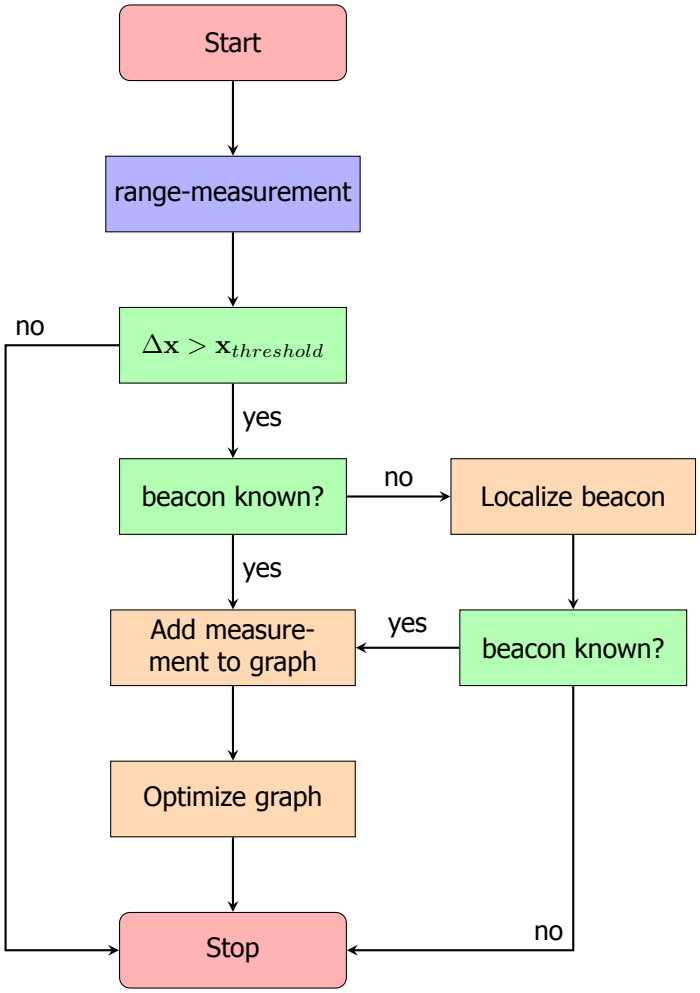


Figure B.2: Flow-diagram of processing an ultra-wideband range-measurement.

type	**what**
transform	$\mathrm{odom} \rightarrow \mathrm{base\_link}$
transform	$\mathrm{base\_link} \rightarrow \mathrm{base\_scan}$
transform	$\mathrm{base\_link} \rightarrow \mathrm{base\_uwb}$
topic	range-measurement stream
topic	scan-observation stream

Table B.1: Dependencies of HRL-SLAM. A transform is a 2D rigid transformation as defined in the ROS-transform library TF2 [95]. Topics are a part of the ROS-implementation of an asynchronous messaging system and can be seen as data-streams to which programs can subscribe.

name	**default value**	**description**
base_frame	"base_link"	name of the base-frame
odom_frame	"odom_link"	name of the odom-frame
laser_frame	"laser_base"	name of the laser-frame
uwb_frame	"uwb_base"	name of the uwb-frame
map_frame	"map"	name of the map-frame
uwb_topic	"uwb_distance"	topic-name where ultra-wideband range mesaurements are published.
lrf_topic	"scan"	topic-name where LiDAR observations are published.

Table B.2: Parameters that specifies frames and data-sources.

name	default value	description
linearUpdate_uwb	1 m	threshold distance before using a new range-measurement
angularUpdate_uwb	0.8 rad	threshold angle before using a new range-measurement
linearUpdate_lrf	1 m	threshold distance before using a new LiDAR-observation
angularUpdate_lrf	0.8 rad	threshold angle before using a new LiDAR-observation
cov_frame_current_pose	"true"	set to true "true" puts prior on most recent pose, otherwise prior is set on the initial pose.
merge_vars	"false"	merge variables if the distance between them is very small, happens often in simulation.
incremental_optim	"5"	amount of variables to be added until optimization of the factor-graph is executed.
incremental_optim	"5"	amount of variables to be needed until optimization of the factor-graph is executed.

Table B.3: Parameters that influence the algorithm flow.

name	default value	description
uwb_sigma	0.3 m	uncertainty of the range measurement
odom_x	0.05 m	odometry uncertainty of the x-direction
odom_y	0.05 m	odometry uncertainty of the y-direction
odom_theta	0.05 rad	odometry uncertainty of the heading
prior_x	0.1 m	prior uncertainty of the x-direction
prior_y	0.1 m	prior uncertainty of the y-direction
prior_theta	0.1 rad	prior uncertainty of the heading

Table B.4: Parameters that influence covariance estimates.

name	default value	description
min_uwb_measurements_for_est	"10"	minimum amount of range measurements used to estimate beacon-location.
circleCircleIntersectionPrecision	"0.001"	absolute tolerance below which we consider touching circles.
SpectralFilterParameter	"10"	minimum ratio difference between the largest and second largest eigenvalue of the measurement matrix needed to trust the beacon-localiation.
useFilteredSet	"true"	option to turn-off the spectral filtering
min_win_ratio	"2"	minimum ratio between the two largest peaks in the voting scheme.
grid_resolution	0.5 m	Width and height of a grid-cell in the voting grid.
force_optimization_for_est	"true"	Force an optimization instance before drawing values from the factor-graph.

Table B.5: Parameters concering beacon-localization.

C

Towards a HRL-SLAM experiment on LEA

Models in general lack many properties of its real equivalent it is meant to describe. A kinematic motion-model, a sensor-observation-model, or any model for that matter, is a compromise between computability, knowledge of the system and to what extent the actual physics is described. There often is no answer to the question if model its shortcomings introduce differences between simulation and reality. The most straightforward manner, in all sciences, to validate the models and answer that question, is to compare the simulation-results to the results using a, real, physical, experiment. The simulation-environment in which HRL-SLAM was tested, contained numerous simplifications of reality. For validation, HRL-SLAM was tested on the Plaza datasets, which contained data from real experiments. However, the Plaza dataset does not contain typical data for which HRL-SLAM was designed for. HRL-SLAM was intended to operate in cluttered indoor environments, which a plaza is definitely not. Secondly, the UWB-radio used in the Plaza datasets has significantly different performance characteristics compared to Decawave's DWM1000 module, which is presumed to be used with HRL-SLAM.

To extend the validation of HRL-SLAM to indoor environments with short-range (compared to the range of the radios in the Plaza datasets) UWB radios, an experiment should be setup using a robot equipped with a DWM1000 module and a set of DWM1000-based beacons in a cluttered indoor environment. This appendix describes the work that has be done towards setting up such an experiment. Due to technical challenges and a limited amount of time, there was no opportunity to finish this work.

The experiment is dependent on three parts to be working:

- HRL-SLAM
- a suitable test-environment
- Embedded software that contains state-machines for both a robot-radio and a beacon-radio

HRL-SLAM is *finished* as in, algorithmically seen. It can be tested and might need minor work given developments on the embedded and communication side. At RCS the testing environments are readily available. Figure C.1 shows the test-apartment at RCS with the robotic stroller Lea Elderly Assistant (LEA). LEA is a robot-stroller that is equipped with a DWM1000-radio and a RPLiDAR. Otherwise the office-environment itself at RCS is could be another suitable test-environment as it is certainly cluttered (desks, chairs) *and* it is significantly larger compared to the apartment.

The part that requires work is the software and hardware on the embedded UWB-systems.

Throughout this thesis the term beacon or radio was used. Hardware-wise, in this case, these terms refer to a system that has a DW1000-module and an antenna which are connected to an ARM-processor. The DW1000 module communicates over a Serial Peripheral Interface bus (SPI) with the ARM-processor. The ARM-system in turn can communicate with a host over RS-485. SPI and RS-485 are both serial communication interfaces (or refer to a standard that describe such an interface).

Figure C.1: The apartment test-environment in the cellar of Taco Scheltemastraat 5, next to the bed stands LEA, with a laptop placed on top of it. Note that although it is a cluttered environment, it is not very big.

A side-note, ranging can only happen between a tag and an anchor. The UWB-radio on the host-machine is generally called the *anchor*. The radios placed in the room are referred to as the *tags*. During this thesis tags have often been called beacons and anchor was usually referred to as *robot* or radio on the robot.

What makes the DW1000-module special is that it has a very accurate clock as it is accurate to picoseconds (10 ps!). This is crucial as the accuracy of the range-calculation is dependent on how well the *time of flight* of the radio waves can be measured. The two-ranging-scheme described in section 2.6 and shown in figure 2.9.

C.1. Notes on Software

Decawave supplies a C-library that can be compiled for the ARM-system as an interface to the DW1000 module. This C-interface supplies straightforward functions that retrieve information such as arrival- and departure-timestamps of radio-messages. The library also supplies the tools necessary to configure the DW1000.

The actual calculation and execution of the two-way-range protocol, and therefor also communication with the DW1000-module, takes places on the ARM-system. Communication with a host machine also takes place on the ARM-system. The software on the ARM-system is usually of the form of an finite-state-machine (FSM) that contains a sub-FSM that executes the two-way-ranging protocol. The system often also includes an interrupt-system that can interrupt the state-machine. For example, the DW1000 can interrupt the ARM-system once it has received a message.

Features such as communication with a host-machine, ranging with multiple beacons *at the same time* or with hard-ware interfaces such as buttons and led-lights are to be programmed on the ARM-system. The two-ranging-scheme itself is also implemented on the ARM-system.

Decawave supplies their code used in demo-products. There are also open-source initiatives that supply such software for specific hardware such as arduinos. Name-worthy open-source projects are the ARM-based Loco Positioning system included with [*] and the arduino-based DW1000 library[†]. There are also closed solutions. What all these projects have in common is that solutions where multiple radios asynchronously range are still under development.

This is why at Robot Care Systems, where there are specific needs for the multiple radios scenarios, their own software for the ARM-system is developed. The remaining part of this appendix will be used to describe the requirements and what has been achieved so far and what contributions were done by

[*] https://github.com/bitcraze/crazyflie-firmware
[†] https://github.com/thotro/arduino-dw1000

me within the scope of this thesis.

C.2. Notes on Hardware

Initially a developer-board (Decwave EKV1000) was used for feasibility checks. This development board was ARM-based and was loaded with demo-software directly from Decawave. The EKV1000 tis shown in figure C.2. Based on the Decawave demo-module, Robot Care Systems developed their own board which included a custom, passive antenna, an ARM-processor and a Decawave DW1000-module. RCS's UWB-software was developed on this first iteration of the RCS-UWB hardware. Due to low-performance, meaning a very low operating-range the board was redesigned to be used with a standard DWM1000-module which is equipped with an active antenna.

Both these radios are shown in figure C.2. To avoid future confusion and to clarify: the DW1000 is the actual chip and the DWM1000 is the combined package of a DW1000-module and an active antenna. Both the DW1000 and DWM1000 are products that can be purchased from Decawave or a Decawave supplier.

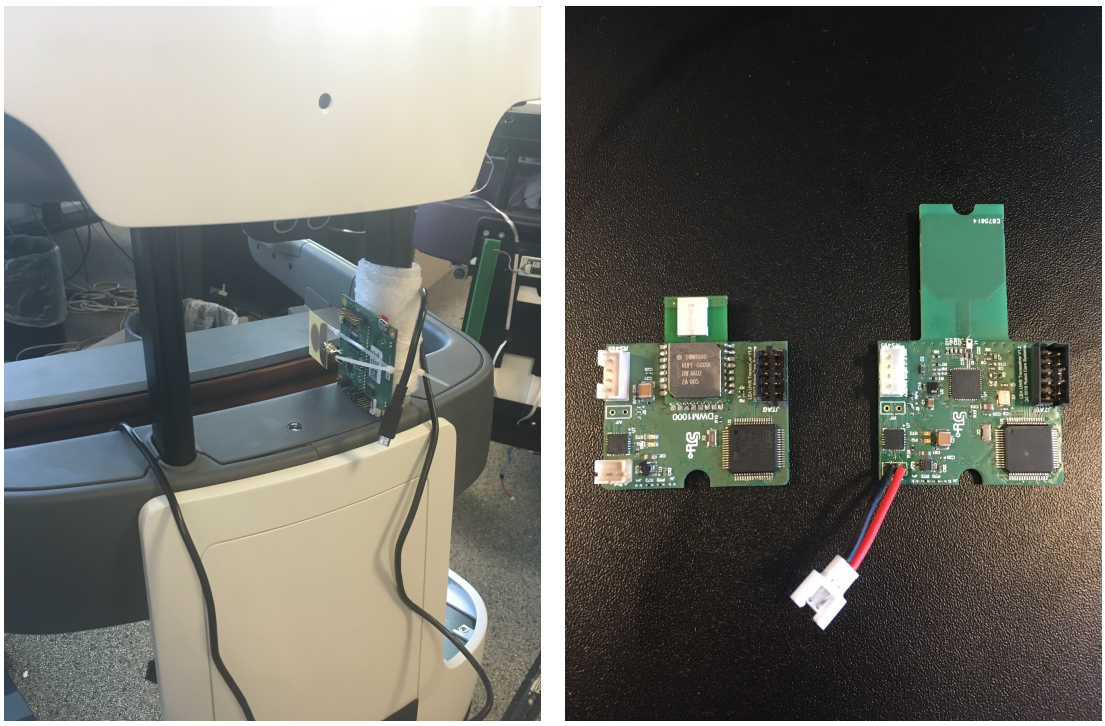


Figure C.2: On the left: Decawave demo-module attached to LEA in a rather primitive matter. On the right: two versions of the ultra-wideband module designed at Robot Care Systems. The left one uses the DWM1000-module with an active antenna, the module on the right is the DW1000 with a passive antenna.

C.3. Ranging with Multiple Radios

For HRL-SLAM it is common that a robot usually observes one or just a few beacons at a time. This means that a robot and the beacons should have a mechanism that allows the robot to range with multiple beacons. Since the ranging-process is dependent on the two-way-ranging scheme, which requires a successful transmission of *5* messages without being interrupted, beacons need a queuing mechanism to make sure each other's ranging procedures are not interrupted all the time.

It should be noted that TDoA-type ranging over radio, especially with the two-way-ranging mechanism is a difficult problem and there is no standard approach to solving it. A recent research proposes

a schema called ATLAS (not to confuse with Bose's hierarchical SLAM framework which is also named ATLAS) which handles multiple tags and one anchor using a centralized server [96]. For RCS however, a solution without such a centralized server is desired.

The first contribution I made to the RCS-UWB software was a simple mechanism that allowed a ranging-session between an anchor and a tag to be interrupted by a second tag. After the interrupt, the anchor would set a range-sequence with the second tag - which would be interpreted by the first tag as an order to be silent. The ability of a tag to distinguish between a ranging-initiation meant for them or an other tag was introduced by adding a *ID* to the header of the ranging-initiation message (a *blink-message* in Decawave terminology).

The problem lies that this method does not scale very well, but in the case of small environments or environments where tags (beacons) are sparsely placed, it can be used. The scalability is limited because any other (random) message that is broadcast by other tags trigger an interrupt rendering the two-way-ranging-scheme erroneous.

It is important to note that if a tag and anchor are ranging, other tags are silenced *unless they want to interrupt*. This requires a scheme the optimizes the network such that all tags get a sufficient amount of ranging. A setup with one anchor and two tags is shown in figure C.3, a video can be seen at*.

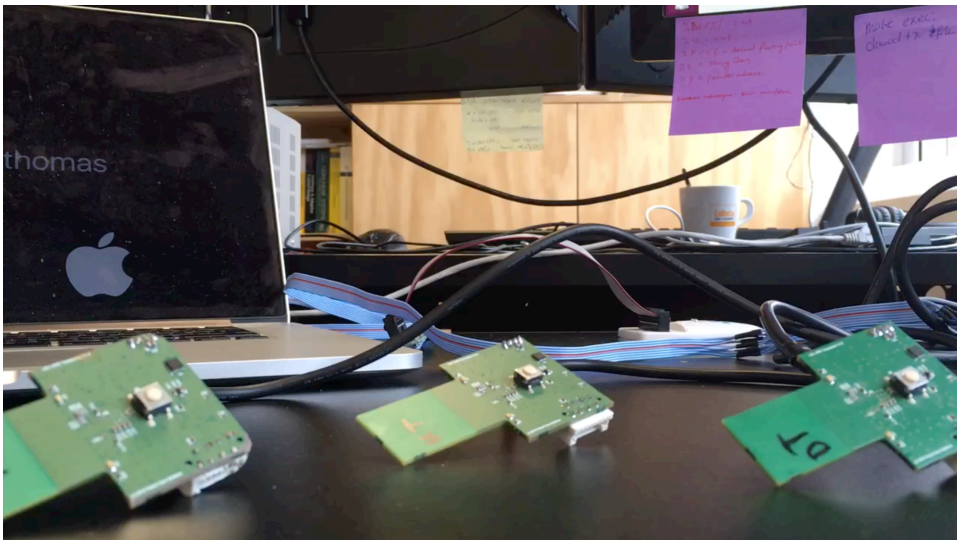


Figure C.3: Setup of two tags and an anchor.

C.4. On DW1000 Configuration & Performance

The performance of the DW1000-module in combination with a passive antenna did not meet the specifications nor the requirements to make it usable for RCS. A solution was first sought into improving the configuration of the DW1000-module prior to upgrading the hardware. Another contribution I made to the RCS-UWB-software was the research performed on finding the optimal settings to improve the operating range.

Automatic Calculation of Duration Parameters

To limit power-usage of the DW1000 modules, the antenna and timestamps calculation can be put on just-in-time schedule. The time it takes for sending a message for instance, can be used to turn the other antenna off to preserve energy. Also the DM1000's internal clock can be timed to turned on for a certain receiving period. These functions are depended on the calculations of certain durations.

*https://www.dropbox.com/s/w2wqsasjw1k60bu/three_uwb.mp4?dl

These time-calculations were initially statically programmed in the RCS-UWB software. However, these timings are configuration-dependent. Selections of preamble-length PRF and other settings used to tweak the performance influence these timings.

In the original Decawave demo-software these timings are derived as in functions depended on the DM1000 configuration. I programmed this principle into the RCS-UWB-software so that in future tweaking of the configuration these timings are derived automatically.

Optimal Configuration for Long Operating-Range Performance

The DW1000-module can be optimized through configuration for high-data flow ($6.8\,\mathrm{MBs}^{-1}$) on short-range or a low-data flow ($110\,\mathrm{kBs}^{-1}$) on long-range. In the case of UWB ranging as required for HRL-SLAM, the interest lies fully into extending the range as we have no special information to send other then a beacon-id.

The primary settings that affect the operating-range and/or data bandwidth are: channel frequency, bandwidth, data-rate, preamble length and Pulse Repetition Frequency (PRF) [97].

The manual recommends to use channel either channel 4 or 7 because they have an extra high bandwidth. $1331.2\,\mathrm{MHz}$ for channel 4 centered around $3993.6\,\mathrm{MHz}$) and $1081.6\,\mathrm{MHz}$ for channel 7 centered around $6489.6\,\mathrm{MHz}$) These channels are only compatible with preamble codes 7 and 8 ($16\,\mathrm{MHz}$ PRF) or preamble codes 17, 18, 19 and 20 ($64\,\mathrm{MHz}$ PRF). it should be noted that the DW1000 is limited to a bandwidth of $900\,\mathrm{MHz}$, this the channel bandwidth difference according to the standard does not apply.

The DW1000 manual recommends a preamble* sequence length of 4096 symbols. The longer the header, the more accurately robust a message can be reconstructed.

C.5. Final Remarks on UWB

At the moment of writing the UWB-software at RCS has reached a level of maturity where it becomes possible to perform at least a simple experiment with one tag and one other anchor. This can already be used to test HRL-SLAM's ability to localize beacons and correct odometry in small environments such as the apartment or a piece of office. Future development of the software should include functions that can alternate ranging between multiple tags. Other new functionalities should extent the software's robustness to radio-interference from other UWB radios that are not part of the system. This is the case for instance in the scenario where two robots try to range with the same set of beacons. These problems all share a solution into embedding more information in the message-payload such that tags and anchors can make an educated decision on when to range and with whom.

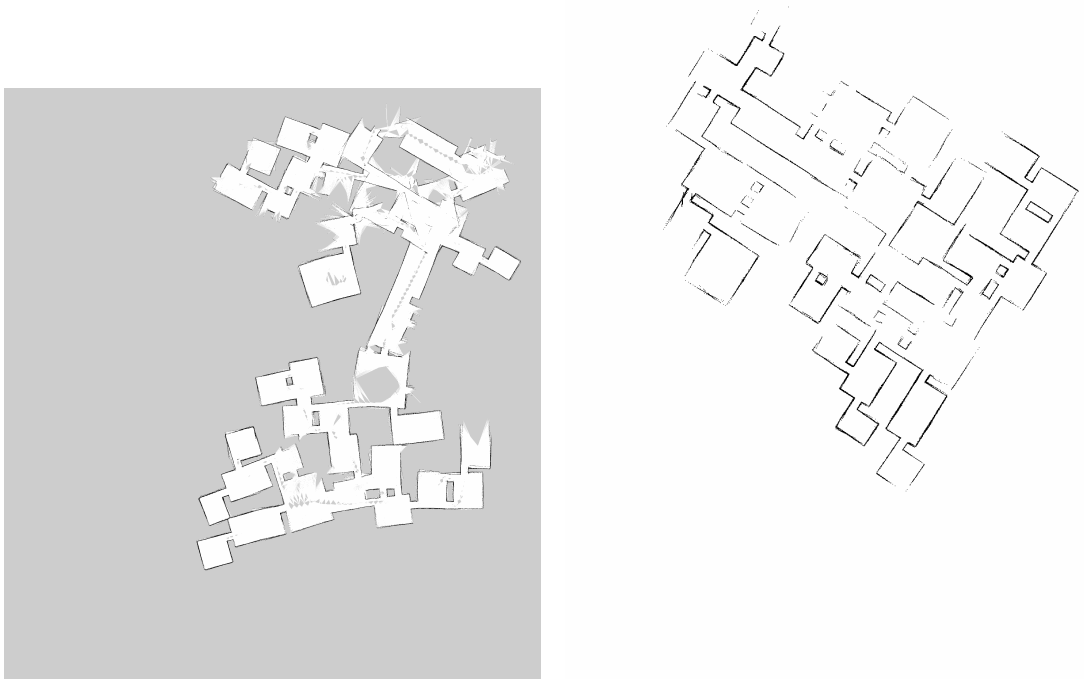
Development of this software was beyond the scope of this thesis, none the less, the experience I gained through working with the UWB system gained insight of realistic refresh-frequencies for sensor-updates, learn typical errors and understand the system beyond the concept of a range. To validate the performance of HRL-SLAM, more time should be put into extending the UWB-software at Robot Care Systems bv - as the result of HRL-SLAM in simulation seems rather impressive.

*which is a sort of header for a radio-message. Through auto-correlation a symbol sequence the message can be decoded

D

Comparison to GMapping

GMapping is a slightly dated Rao-Blackwellized Particle Filter SLAM-algorithm [19]. However, it is still used successfully on many occasions. GMappings makes use of an ego-motion estimate from odometry and uses LiDAR scans to map and use scan to scan matching to correct the pose-trajectory. The GMapping application has been run on the Maze75x75 experiment to do compare the results to the proposed HRL-SLAM-algorithm. As expected, it becomes very difficult for GMapping to correct long-term drifts in large loops using solely a laser-scan in such a large and ambigious environment. It should be noted that the comparison is unfair in a sense that GMapping does not use the same information as HRL-SLAM (range-measurements are ignored). Secondly, it uses a dated filter based-approach. Yet, the principle is demonstrated: if a low-spec LiDAR is used, it might be better to do pose-correction using UWB-range-measurements.



(a) Result using GMapping.

(b) Result using HRL-SLAM.

Figure D.1: Comparison of GMapping and HRL-SLAM in the Maze75x75 experiment. Note that GMapping does not use the range-measurements. Also, note that GMapping produces a true occupancy grid-map with free spaces (white), while HRL-SLAM produces grid-map that does not differentiate between free-space and unexplored space.

Bibliography

[1] 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.

[2] M. Cakmak and L. Takayama, "Towards a comprehensive chore list for domestic robots," in *Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction*, pp. 93–94, IEEE Press, 2013.

[3] A. Davids, "Urban search and rescue robots: from tragedy to technology," *IEEE Intelligent systems*, vol. 17, no. 2, pp. 81–83, 2002.

[4] E. B. Olson, "Robust and efficient robotic mapping," 2008.

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

[6] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[7] C. Zhao, W. Pan, and H. Hu, "Interactive indoor environment mapping through visual tracking of human skeleton," *International Journal of Modelling, Identification and Control*, vol. 20, no. 4, pp. 319–328, 2013.

[8] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *Intl. J. of Robotics Research (IJRR)*, vol. 31, pp. 217–236, Feb. 2012.

[9] G. Grisetti, R. Kümmerle, C. Stachniss, U. Frese, and C. Hertzberg, "Hierarchical optimization on manifolds for online 2d and 3d mapping," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 273–278, IEEE, 2010.

[10] V. Ila, L. Polok, M. Solony, and P. Svoboda, "Highly efficient compact pose SLAM with SLAM++," *CoRR*, vol. abs/1608.03037, 2016.

[11] M. F. E. Rohmer, S. P. N. Singh, "V-rep: a versatile and scalable robot simulation framework," in *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[12] J. Djugash, B. Hamner, and S. Roth, "Navigating with ranging radios: Five data sets with ground truth," *Journal of Field Robotics*, vol. 26, no. 9, pp. 689–695, 2009.

[13] A. Elfes, "Occupancy grids: a probabilistic framework for robot perception and navigation.," 1991.

[14] S. Huang and G. Dissanayake, "A critique of current developments in simultaneous localization and mapping," *International Journal of Advanced Robotic Systems*, vol. 13, no. 5, p. 1729881416669482, 2016.

[15] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[16] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, *et al.*, "Fastslam: A factored solution to the simultaneous localization and mapping problem," in *Aaai/iaai*, pp. 593–598, 2002.

[17] A. Doucet, N. De Freitas, K. Murphy, and S. Russell, "Rao-blackwellised particle filtering for dynamic bayesian networks," in *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pp. 176–183, Morgan Kaufmann Publishers Inc., 2000.

[18] M. Montemerlo and S. Thrun, "Fastslam 2.0," *FastSLAM: A scalable method for the simultaneous localization and mapping problem in robotics*, pp. 63–90, 2007.

[19] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, pp. 34–46, Feb 2007.

[20] S. J. Julier and J. K. Uhlmann, "A counter example to the theory of simultaneous localization and map building," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 4, pp. 4238–4243, IEEE, 2001.

[21] F. R. Kschischang, B. J. Frey, and H. A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, pp. 498–519, Feb 2001.

[22] F. Dellaert and M. Kaess, "Square root sam: Simultaneous localization and mapping via square root information smoothing," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 2006.

[23] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*, pp. 3607–3613, May 2011.

[24] F. Dellaert, "Factor graphs and gtsam: A hands-on introduction," tech. rep., Georgia Institute of Technology, 2012.

[25] S. Agarwal, K. Mierle, and Others, "Ceres solver." http://ceres-solver.org.

[26] D. M. Rosen, M. Kaess, and J. J. Leonard, "An incremental trust-region method for robust online sparse least-squares estimation," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 1262–1269, IEEE, 2012.

[27] H. Wang, S. Huang, U. Frese, and G. Dissanayake, "The nonlinearity structure of point feature slam problems with spherical covariance matrices," *Automatica*, vol. 49, no. 10, pp. 3112–3119, 2013.

[28] L. Carlone, R. Aragues, J. A. Castellanos, and B. Bona, "A fast and accurate approximation for planar pose graph optimization," *The International Journal of Robotics Research*, vol. 33, no. 7, pp. 965–987, 2014.

[29] K. Khosoussi, S. Huang, and G. Dissanayake, "Exploiting the separable structure of slam," in *Robotics: Science and systems*, 2015.

[30] E. Olson, J. Leonard, and S. Teller, "Fast iterative alignment of pose graphs with poor initial estimates," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 2262–2269, IEEE, 2006.

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

[32] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 2, pp. 1322–1328, IEEE, 1999.

[33] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *Artificial intelligence*, vol. 128, no. 1-2, pp. 99–141, 2001.

[34] A. F. Smith and A. E. Gelfand, "Bayesian statistics without tears: a sampling–resampling perspective," *The American Statistician*, vol. 46, no. 2, pp. 84–88, 1992.

[35] D. Wai, "Visualization of resampling in sequential monte carlo methods." http://denniswai.com/research/. [Online; accessed July 16, 2017], modified for report.

[36] N. Sünderhauf and P. Protzel, "Switchable constraints vs. max-mixture models vs. rrr-a comparison of three approaches to robust pose graph slam," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 5198–5203, IEEE, 2013.

[37] E. Olson and P. Agarwal, "Inference on networks of mixtures for robust robot mapping," *The International Journal of Robotics Research*, vol. 32, no. 7, pp. 826–840, 2013.

[38] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard, "Robust map optimization using dynamic covariance scaling," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 62–69, IEEE, 2013.

[39] Y. Latif, C. Cadena, and J. Neira, "Robust loop closing over time for pose graph slam," *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1611–1626, 2013.

[40] G. Agamennoni, P. Furgale, and R. Siegwart, "Self-tuning m-estimators," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 4628–4635, IEEE, 2015.

[41] H. Kretzschmar, C. Stachniss, and G. Grisetti, "Efficient information-theoretic graph pruning for graph-based slam with laser range finders," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 865–871, IEEE, 2011.

[42] L. Carlone, Z. Kira, C. Beall, V. Indelman, and F. Dellaert, "Eliminating conditionally independent sets in factor graphs: A unifying perspective based on smart factors," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 4290–4297, IEEE, 2014.

[43] N. Carlevaris-Bianco, M. Kaess, and R. M. Eustice, "Generic node removal for factor-graph slam," *IEEE Transactions on Robotics*, vol. 30, no. 6, pp. 1371–1385, 2014.

[44] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner, "On measuring the accuracy of slam algorithms," *Autonomous Robots*, vol. 27, no. 4, pp. 387–407, 2009.

[45] Wikipedia, "Mahalanobis distance," 2017. [Online; accessed 18-July-2017].

[46] D. Fox, "Kld-sampling: Adaptive particle filters," in *Advances in neural information processing systems*, pp. 713–720, 2001.

[47] J. Knuth and P. Barooah, "Error growth in position estimation from noisy relative pose measurements," *Robotics and Autonomous Systems*, vol. 61, no. 3, pp. 229–244, 2013.

[48] P. W. Thorndyke and B. Hayes-Roth, "Differences in spatial knowledge acquired from maps and navigation," *Cognitive psychology*, vol. 14, no. 4, pp. 560–589, 1982.

[49] T. P. McNamara, "Spatial representation," *Geoforum*, vol. 23, no. 2, pp. 139–150, 1992.

[50] P. W. Thorndyke, "Distance estimation from cognitive maps," *Cognitive psychology*, vol. 13, no. 4, pp. 526–550, 1981.

[51] A. Stevens and P. Coupe, "Distortions in judged spatial relations," *Cognitive psychology*, vol. 10, no. 4, pp. 422–437, 1978.

[52] A. Pronobis, O. Martinez Mozos, B. Caputo, and P. Jensfelt, "Multi-modal semantic place classification," *The International Journal of Robotics Research*, vol. 29, no. 2-3, pp. 298–320, 2010.

[53] H. Zender, P. Jensfelt, O. M. Mozos, G.-J. M. Kruijff, and W. Burgard, "An integrated robotic system for spatial understanding and situated interaction in indoor environments," in *AAAI*, vol. 7, pp. 1584–1589, 2007.

[54] B. Roskos-Ewoldsen, T. P. McNamara, A. L. Shelton, and W. Carr, "Mental representations of large and small spatial layouts are orientation dependent.," *Journal of Experimental Psychology: Learning, Memory, and Cognition*, vol. 24, no. 1, p. 215, 1998.

[55] O. Martinez Mozos, A. Rottmann, R. Triebel, P. Jensfelt, W. Burgard, *et al.*, "Semantic labeling of places using information extracted from laser and vision sensor data," 2006.

[56] A. Pronobis and P. Jensfelt, "Large-scale semantic mapping and reasoning with heterogeneous modalities," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 3515–3522, IEEE, 2012.

[57] Blippar, "Blippar." https://blippar.com/en/products/computer-vision-api/. [Online; accessed July 24, 2017].

[58] M. Bosse, P. Newman, J. Leonard, and S. Teller, "Simultaneous localization and map building in large-scale cyclic environments using the atlas framework," *The International Journal of Robotics Research*, vol. 23, no. 12, pp. 1113–1139, 2004.

[59] J.-L. Blanco, J.-A. Fernández-Madrigal, and J. Gonzalez, "A new approach for large-scale localization and mapping: Hybrid metric-topological slam," in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 2061–2067, IEEE, 2007.

[60] M. A. Paskin, *Thin Junction Tree Filters for Simultaneous Localization and Mapping*. Citeseer, 2002.

[61] M. Kaess, V. Ila, R. Roberts, and F. Dellaert, "The bayes tree: An algorithmic foundation for probabilistic robot mapping," in *Algorithmic Foundations of Robotics IX*, pp. 157–173, Springer, 2010.

[62] Decawave, "Decawave technology hits a home run at microsoft indoor localization competition." http://www.marketwired.com. [Online; accessed August 8, 2017].

[63] M. J. Segura, F. A. Auat Cheein, J. M. Toibero, V. Mut, and R. Carelli, "Ultra wide-band localization and slam: A comparative study for mobile robot navigation," *Sensors*, vol. 11, no. 2, pp. 2035–2055, 2011.

[64] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of wireless indoor positioning techniques and systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 6, pp. 1067–1080, 2007.

[65] Sewio, "Two-way-ranging." [Online; accessed February 20, 2017].

[66] P. Newman and J. Leonard, "Pure range-only sub-sea slam," in *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 2, pp. 1921–1926, IEEE, 2003.

[67] E. Olson, J. J. Leonard, and S. Teller, "Robust range-only beacon localization," *IEEE Journal of Oceanic Engineering*, vol. 31, no. 4, pp. 949–958, 2006.

[68] J.-L. Blanco, J. González, and J.-A. Fernández-Madrigal, "A pure probabilistic approach to range-only slam," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pp. 1436–1441, IEEE, 2008.

[69] A. Kehagias, J. A. Djugash, and S. Singh, "Range-only slam with interpolated range data," 2006.

[70] B. Boots and G. Gordon, "A spectral learning approach to range-only SLAM," in *Proceedings of the 30th International Conference on Machine Learning* (S. Dasgupta and D. McAllester, eds.), vol. 28 of *Proceedings of Machine Learning Research*, (Atlanta, Georgia, USA), pp. 19–26, PMLR, 17–19 Jun 2013.

[71] R. Shariff, A. György, and C. Szepesvári, "Exploiting symmetries to construct efficient mcmc algorithms with an application to slam," in *Artificial Intelligence and Statistics*, pp. 866–874, 2015.

[72] P. Torma, A. György, and C. Szepesvári, "A markov-chain monte carlo approach to simultaneous localization and mapping," in *International conference on artificial intelligence and statistics*, pp. 852–859, 2010.

[73] E. Takeuchi, A. Elfes, and J. Roberts, "Localization and place recognition using an ultra-wide band (uwb) radar," in *Field and Service Robotics*, pp. 275–288, Springer, 2015.

[74] L. Génevé, A. Habed, É. Laroche, O. Kermorgant, and E. C. d. N. LS2N, "Robust range-only mapping via sum-of-squares polynomials," 2017.

[75] F. Herranz, Á. Llamazares, E. Molinos, and M. Ocaña, "A comparison of slam algorithms with range only sensors," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 4606–4611, IEEE, 2014.

[76] V. Indelman, S. Williams, M. Kaess, and F. Dellaert, "Information fusion in navigation systems via factor graph based incremental smoothing," *Robotics and Autonomous Systems*, vol. 61, no. 8, pp. 721–738, 2013.

[77] Q. Zeng, W. Chen, J. Liu, and H. Wang, "An improved multi-sensor fusion navigation algorithm based on the factor graph," *Sensors*, vol. 17, no. 3, p. 641, 2017.

[78] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.

[79] Y. Zhou, "An efficient least-squares trilateration algorithm for mobile robot localization," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3474–3479, Oct 2009.

[80] A. Prorok, L. Gonon, and A. Martinoli, "Online model estimation of ultra-wideband tdoa measurements for mobile robot localization," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 807–814, Ieee, 2012.

[81] D. M. Rosen, M. Kaess, and J. J. Leonard, "Robust incremental online inference over sparse factor graphs: Beyond the gaussian case," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 1025–1032, IEEE, 2013.

[82] G. Hu, K. Khosoussi, and S. Huang, "Towards a reliable slam back-end," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pp. 37–43, IEEE, 2013.

[83] C. Ding, X. He, H. Zha, M. Gu, and H. D. Simon, "A minmaxcut spectral method for data clustering and graph partitioning," *Lawrence Berkeley National Laboratory, Tech. Rep*, vol. 54111, 2003.

[84] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[85] V. Y. Pan and Z. Q. Chen, "The complexity of the matrix eigenproblem," in *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pp. 507–516, ACM, 1999.

[86] M. Foltin, *Automated maze generation and human interaction*. PhD thesis, Masarykova univerzita, Fakulta informatiky, 2011.

[87] D. Ltd, *DWM1000 IEEE 802.15.4-2011 UWB Transceiver Module*. Decawave Ltd.

[88] J. González, J.-L. Blanco, C. Galindo, A. Ortiz-de Galisteo, J.-A. Fernandez-Madrigal, F. A. Moreno, and J. L. Martínez, "Mobile robot localization based on ultra-wide-band ranging: A particle filter approach," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 496–507, 2009.

[89] G. Bellusci, "Ultra-wideband ranging for low-complexity indoor positioning applications," 2011.

[90] V. Dhiman, A. Kundu, F. Dellaert, and J. J. Corso, "Modern map inference methods for accurate and fast occupancy grid mapping on higher order factor graphs," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 2037–2044, IEEE, 2014.

[91] J. Lee, K. Li, S. Zlatanova, T. Kolbe, C. Nagel, and T. Becker, "Ogc® indoorgml," *Open Geospatial Consortium*, 2014.

[92] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, p. 5, Kobe, 2009.

[93] P. Fankhauser and M. Hutter, "A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation," in *Robot Operating System (ROS) – The Complete Reference (Volume 1)* (A. Koubaa, ed.), ch. 5, Springer, 2016.

[94] G. Guennebaud, B. Jacob, *et al.*, "Eigen v3." http://eigen.tuxfamily.org, 2010.

[95] T. Foote, "tf: The transform library," in *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, Open-Source Software workshop, pp. 1–6, April 2013.

[96] J. Tiemann, F. Eckermann, and C. Wietfeld, "Atlas-an open-source tdoa-based ultra-wideband localization system," in *Indoor Positioning and Indoor Navigation (IPIN), 2016 International Conference on*, pp. 1–6, IEEE, 2016.

[97] D. Ltd, *DW1000 IEEE 802.15.4-2011 UWB Transceiver Module*. Decawave Ltd.