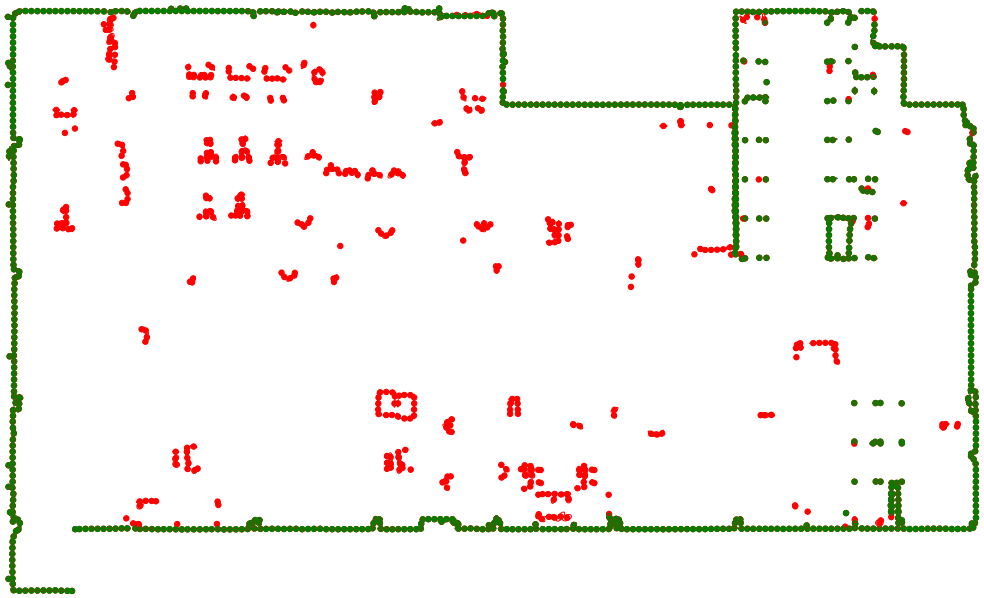
CHALMERS
UNIVERSITY OF TECHNOLOGY



Long-term mapping and localization of industrial environments using LiDAR

FELIX MILLQVIST
JAKOB VINKÅS

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021
www.chalmers.se

Master's thesis 2021

Long-term mapping and localization of industrial environments using LiDAR

FELIX MILLQVIST
JAKOB VINKÅS

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2021

Long-term mapping and localization
of industrial environments using LiDAR
FELIX MILLQVIST JAKOB VINKÅS

Supervisor: Sonnie Hallberg, Soft Design RTS
Examiner: Lars Hammarstrand, Electrical Engineering

Master's Thesis 2021
Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Dual-time normal distributions transform map.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2021

Long-term mapping and localization of industrial environments using LiDAR
FELIX MILLQVIST, JAKOB VINKÅS
Electrical Engineering
Chalmers University of Technology

Abstract

Automated Guided Vehicles (AGV's) are widely used for transporting material in industrial environments. The AGV's typically localize using reflectors manually mounted in the environment to serve as known reference points for localization. By positioning a set of reflectors using the onboard LiDAR, the AGV can triangulate its position in the environment. Mounting the reflectors is expensive, removing the need for reflectors and instead rely on the natural features of the environment would greatly reduce cost and increase the flexibility of route planning.

This thesis aims to provide a system able to replace the currently used reflector-based system using the natural features of the environment. More specifically the thesis aims to implement a system able to create a map from no prior knowledge using the existing open-source system Google Cartographer. Convert the produced map into a Normal Distributions Transform (NDT) structure to increase localization accuracy, and use a particle filter for localization.

A common challenge when localizing using the natural features of the environment is handling dynamic environments with moving objects and when the environment is drastically changed such walls being obscured by other large objects. This thesis will investigate the use of a Dual-Timescale NDT (DT-NDT) to improve the localization by utilizing the current state of the environment and not only static objects such as walls.

Two main tests were conducted to compare the reflector-based system and the new system. In the first test, the environment was not drastically changed, however, it was still dynamic due to multiple people and other AGV's moving around. In the second test, the environment was drastically changed by introducing a new wall. In the first test, the new system showed similar results to the reflector-based system even though the reflectors were not used. In the second test, the reflector-based system failed to localize in the environment while the new system was able to localize. The new system performed better when the environment was not drastically changed, however, it was still accurate enough to be used in production. Localization with no prior knowledge is possible with the new system but is not robust enough for production use.

Keywords: localization, NDT, dual-timescale, particle filter, google cartographer.

Acknowledgements

First of all we would like to thank our academic supervisor Lars Hammarstrand for his support and guidance. We would also like to thank Sonnie Hallberg who was our supervisor from the company Soft Design, and the entire staff at Soft Design for their support and help in times of need.

Felix Millqvist, Jakob Vinkås, Gothenburg, May 2021

Contents

List of Figures

List of Tables

List of Algorithms

1

Introduction

Automated Guided Vehicles, (AGV's) have been used for transporting material and products in industrial environments such as production areas and warehouses for many years [1]. There are multiple ways of navigating AGV's that have been developed throughout the years such as magnetic strips or rails in the floor that guides the AGV towards the target location [2]. Another, more flexible way of navigating the AGV's is to rely on the ability to localize in the environment while not having any physical traces to follow such that the routes can be easily changed without making any physical changes to the environment. Typically, the localization problem is solved by using reflectors that are strategically placed in the environment to serve as known reference points [3]. The positions of the reflectors are measured by onboard sensors and a reference map consisting of the positions of the reflectors is created. During operation, a Light Detection And Ranging (LiDAR) sensor measures the angles and distances to the reflectors from the AGV, and by comparing the measurements to the reference map an estimation of the position of the AGV can be made using triangulation. The downside of mounting and manually measuring the positions of reflectors throughout the environment is that it is expensive and time-consuming. In some environments such as tight gates, it can also be difficult to fit the reflectors. To reduce cost and time consumption, a way to localize without the need for reflectors or to make any other physical changes to the environment needs to be developed.

The company Soft Design currently provides AGV's to their consumers that use reflectors for localization and this work is conducted in cooperation with the company. In addition to a single-layer LiDAR sensor, the company also uses wheel encoders as odometry to aid the localization. The localization that the company currently uses is adequately accurate and robust for long-term use and will therefore be used as a comparison. While the accuracy of the localization is highly dependent on the number of reflectors visible at any given time and the accuracy of the reference map the currently used system is in general able to estimate the position of the LiDAR sensor with an accuracy of around ± 5 mm at lower velocities.

To localize in an environment without making any changes to it, the already existing features of the environment must be used. A potential solution is to imitate the reflector method of localization by finding natural landmarks, such as pillars and corners, in the environment and similarly use these landmarks as the reflectors are

used [4, 5]. However, finding such natural landmarks is a nontrivial task since they need to be stationary and detectable for long-term use. Industrial environments are constantly changing due to moving objects such as robots, humans, and forklifts. Such objects could potentially be discovered during a single mapping of the environment, however, there are also semi-static objects that would not be detected during a single mapping, for example, pallets, non-operating robots, and more.

Objects that are partially fixed in place such as pallets could not only be miss interpreted as static objects but can also obscure the view of the actual static objects such as walls, corners, pillars, etc. To reach an accurate and robust localization, the reference map must be as detailed and accurate as possible. Semi-static objects could therefore present an issue for localization over a longer period of time since they might be removed or slightly moved from one-time instance to another.

1.1 Related Work

There already exist algorithms that can solve the simultaneous localization and mapping (SLAM) problem with varying accuracy and robustness. In [6], three different algorithms, namely Google Cartographer [7], GMapping [8] , and tinySLAM [9] are tested and compared. It is concluded that all of these algorithms seem to be able to generate a trajectory that is quite accurate compared to the ground truth. TinySLAM is developed to be a very simple SLAM system and uses an occupancy grid map with a single hypothesis for the world state represented as a probability density function (PDF). TinySLAM uses a Monte-Carlo method to match the map and the sensor data. Due to the simplicity of the algorithm, it has a lower accuracy compared to the other two algorithms and it also lacks the ability to perform loop-closures. GMapping uses the basic idea of TinySLAM. However, it uses multiple hypotheses for the position and heading of the robot and a Rao-Blackwellized particle filter to approximate the PDF to be able to change the world state more effectively and find loop closure and a more accurate trajectory. Google Cartographer uses a graph-based occupancy grid where the vertices contain submaps in the form of grid maps and the edges are the transitions between the submaps. Cartographer uses a brute-force method to optimize the matching which makes the algorithm slower than GMapping and TinySLAM, however, the accuracy is often better. In regards to accuracy, based on the findings made in [6], none of the algorithms are capable of achieving the accuracy required in this work, however, the creation of a map could still be utilized.

All of the algorithms evaluated in [6] used an occupancy grid map structure. However, there are many other ways to structure a map to be able to use it for localization. The Normal Distributions Transform (NDT) map structure has been discussed in multiple articles [10, 11, 12]. While there are ways to create a map from no prior knowledge using the NDT map structure [11, 12], no sources were found that evaluated the localization using a map that was automatically generated using a SLAM algorithm based on the NDT structure. In [13], Google Cartographer was used to derive an initial map of the environment. The occupancy grid map gen-

erated by Google Cartographer was then converted into the NDT structure. The NDT map was then used with an already implemented localization algorithm and the map was never updated. This map structure showed promising results both in accuracy and computation complexity. The work was done in an environment with the same characteristics as the environments addressed in this work. However, the localization algorithm was already implemented and used by the company the work was conducted with.

Multiple articles have researched localizing using an NDT map. However, they usually have prior knowledge about the environment and build the map manually [14, 15]. Localization in a known environment using the NDT map structure in combination with a Monte Carlo Localization filter, also referred to as a particle filter was evaluated in [14, 15]. The articles showed promising results in terms of both accuracy and robustness. In [15], the use of a Dual Time-scale (DT) map was introduced to handle dynamic environments in a more robust way, namely a static and a dynamic map. In [15] it was suggested that the added dynamic map should keep track of the current state of the environment by continuously updating the map and thereby enable the use of semi-static objects during localization. The authors suggested that the structure for the dynamic map should be a hybrid between NDT and occupancy grid as presented in [16] such that the probability of a cell being occupied would be taken into consideration. However, the static map was manually created prior to any testing and was never updated as the environment changed over time.

1.2 Objective and Method

The company, Soft Design, currently uses AGV's equipped with a localization system consisting of a single-layer LiDAR sensor in combination with multiple reflectors mounted in the environment with the addition of wheel encoders. The future vision is to be able to replace the current reflector-based localization system and to localize without any additions or modifications to the environment. The system developed, during this work, will therefore be compared to the current system to be able to evaluate the performance. Note that while the current reflector-based system has an accuracy of ± 5 mm in optimal circumstances, many of the applications allow for a slightly larger error.

The work in this thesis has created an initial map using an existing occupancy grid SLAM implementation and converted the map to an NDT structure [10, 11, 12] as described in [13]. The evaluation of an observation will be based on the proposed method found in [10, 11, 12] while using the particle filter for localization as described in [14, 15]. To be able to use the semi-static objects in the environment, the dual time-scale map structure introduced in [14, 15] is used. The dynamic map in the dual time-scale map is updated using a combination of the suggested methods provided in [16, 17, 18]. A proposal of how the static map in the dual time-scale map can be updated using the dynamic map from separate time instances has also been provided in this work. In conclusion, the work will create and implement a complete system

to be able to automatically generate a map from no prior knowledge, update the map over time, and use the map for localization with sufficient accuracy to be used in production.

1.3 Limitations

In this work, the localization of the AGV will be investigated and developed. Many sources use the term navigation when discussing similar tasks, however, navigation also involves path planning and collision avoidance. Since the main focus of the work is how to handle semi-static objects, some of the design choices made that are not directly relevant to the main focus will be made based on limited research. In the proposed method the initial map will be derived using a preexisting open-source solution, the initial research showed that Google Cartographer shows promising results in regards to accuracy of the derived map while being able to handle loop-close better than the other available open-source SLAM implementations. The final goal is to be able to use the implemented system on the AGV:s provided by the company Soft Design. However, for the duration of this work, the computational complexity and hardware requirements in regards to computational requirements will not be prioritized. The physical position of the sensors has been delimited. The placement of the sensors will be estimated with different measurements but not optimized. This could potentially lead to a skewed trajectory when a turn is taken.

1.4 Contributions and Main Results

This thesis was able to reach the following main results and conclusion:

- An NDT map could sufficiently be constructed by converting the occupancy grid provided by the open-source software Google Cartographer
- A way to automatically remove undesirable semi-static objects from the static map was presented with the desired outcome.
- It was proven that for the test cases chosen, the proposed DT-NDT system implemented in this thesis was accurate enough to replace the currently used reflector-based system when a prior pose of the AGV was known.
- It was also found that the dynamic map in the DT-NDT map structure helped reduce the error introduced in the localization when drastic changes where made to the environment. In comparison to the NDT map structure with only a static map.

2

Background

To localize in an environment with sufficient accuracy, a map must be used as a frame of reference. This map must be of the correct structure and accurate enough to enable localization with sufficient accuracy. This is no simple task and to solve it in this work, information, implementations and findings collected from multiple sources will be used. All of these preliminaries will be explained in this section. It is also important to understand the mechanical properties of the AGV used for all testing and this will also be explained in this Section.

2.1 System Setup

In this section, the setup for this work will be described. The AGV that was used in this project is equipped with wheel-encoders, a new LiDAR sensor and an old LiDAR sensor. The old LiDAR is bought with a system that localizes using reflectors and the system can therefore not be analyzed on a implementation level. The new LiDAR sensor will be used for testing while the old LiDAR sensor will be used to compare the results and evaluate the performance of the proposed implementation. The AGV is designed to be used in industrial environments and warehouses. These environments are constantly changing which needs to be addressed to have a robust state prediction whenever the AGV is used. The data used will be real sensor data from the sensors on the AGV and the environment that will be used is the production and warehouse of the company Soft Design.

2.1.1 Current Localization System

The current AGV has a localization system based on reflectors with a LiDAR. The LiDAR is bought with pre-developed software for localizing. The LiDAR has to see a number of reflectors to triangulate where in the environment the AGV is. The reflectors are installed by the company in a way similar to when a system is deployed at a customer location. The current LiDAR has a frequency of 8 Hz and measures 1440 points per revolution.

2.1.2 AGV Coordinate System

The AGV has two internal coordinate systems since the LiDAR sensor and the wheelbase has an offset in the x-axis and one global map coordinate system as can be seen in Figure 2.1.

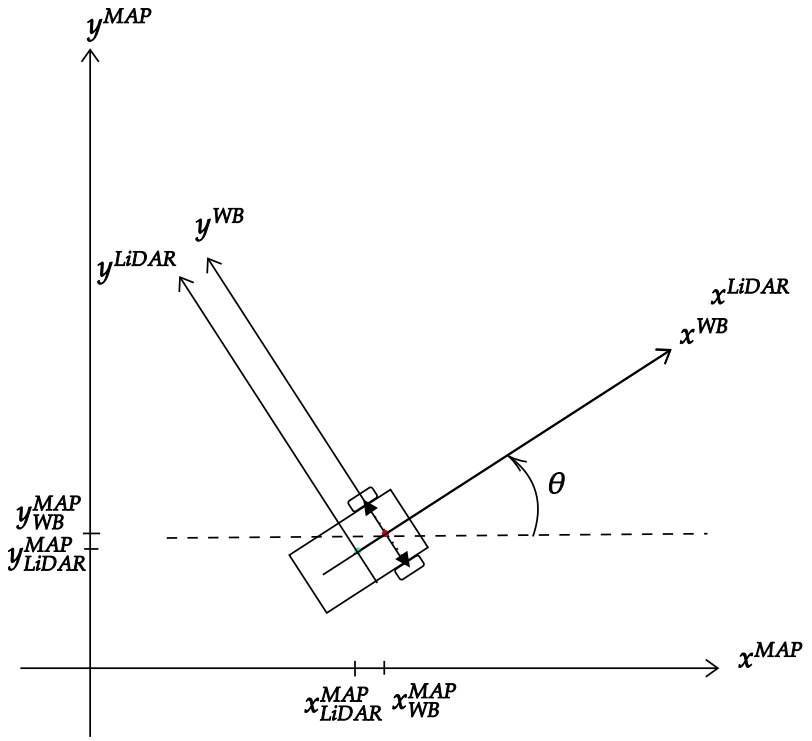


Figure 2.1: Wheel base frame vs LiDAR frame

2.1.3 LiDAR Sensor

A LiDAR is a measuring device that uses laser light to measure the distance from the sensor to an object. While the LiDAR sensor has been around for a long time [19, 20], the accuracy that it is nowadays able to provide makes it a common sensor to use for localization when trying to solve similar problems, namely localizing in a known restricted area.

The LiDAR sensor used in this work is a high-end industrial sensor, the HD R2000 from Pepperl and Fuchs. The LiDAR is capable of producing up to 84000 measurements per second in different configurations. The configuration that was used throughout the entire thesis was to set the LiDAR to produce 2400 measurements per revolution at 35 revolutions per second. The R2000 LiDAR is not only capable of measuring the distance to an object but also the amount of reflectivity of the object by measuring the intensity of the light source that reflects back to the sensor. This is the reason why reflector-based solutions are used in the previous solution to the localization problem since they can both be detected and the distance to them can be derived from the data given by the LiDAR sensor. Note that the existing system does not use the exact same LiDAR sensor but instead uses the SICK NAV350

unit with integrated localization, however, even though the specification of the two sensors differ, the capability to measure both the distance to- and reflectivity of- an object is similar for both of the sensors.

2.1.4 Robot Operating System

Robot Operating System (ROS) is an open-source software framework that is built for robotics. The framework simplifies the structure between different parts of the application with the help of modular design. ROS provides a way to build modules that connects to become an application. There are two types of modules, also called nodes, publisher and subscriber [21]. The publisher publishes some data on a so-called topic. The topic behaves like a bus where data messages are. The subscriber is the opposite of the publisher, it subscribes to a topic to receive the data message that has been published. In this way, it is easy to build small modules that only do a specific task. ROS also provides a convenient way to store data, called rosbag. Rosbags stores the data on the topics that are published. It is possible to replay the rosbag later while the data is given such that the application thinks the data comes from the nodes.

2.1.5 Computer Hardware Setup and Interface

In this work, an external laptop will be running the algorithm and communicate with the AGV. The laptop is an HP Elitebook with a Intel® Core i7-8565U CPU @ 1.80GHz x 8 with 16Gb memory. The laptop is running Ubuntu 20.04.2 LTS. The communication with the AGV is done with a serial port where odometry is received and the pose estimate is sent. The LiDAR is connected to the ethernet port and communicates with HTML requests.

2.2 Simultaneous localization and mapping

SLAM is the concept of being able to localize in an environment without any prior knowledge of the environment and simultaneously map the environment. The map can then be used as it is being updated in order to constantly keep track of the vehicles position on the map while improving not only the map but potentially also the accuracy of the localization. Regardless of the sensor setup, the basis for the solution is to use the data given by the sensors and the map must therefore be constructed in a way that future measurement can be used in combination with the map.

In order to localize in an environment, an initial map must be created. In [6] a number of existing open-source implementations for solving the problem were tested. In the article it was shown that rather simple implementation is sufficient when the demand for accuracy in the derived map is low, using for example tinySLAM [9]. However, solving the problem of skewed maps and failed loop closures during mapping is difficult. Loop-closure is when the AGV arrives to the same position from a different route, the system needs to recognize that it is the same position.

This needs to be solved for the AGV to localize an arbitrary route. In [6, 13] it was concluded that Google Cartographer [7] gave a robust and accurate mapping while also sufficiently solving the loop closure problem.

The most common map structure for the SLAM procedure is to use an occupancy grid as described in Section 2.3 [7, 8, 9]. It is, however, not the only map structure suitable for a SLAM procedure. Another common conceptual map structure is to look for distinguishable landmarks in the environment, similar to the reflector based system described in Section 2.1.1, the landmarks are matched with landmarks on the reference map and the pose of the sensor is derived using triangulation. While it is possible to use this map structure, most sources present it as more of a conceptual and easy to understand SLAM procedure as very few are able to reach a high accuracy. It is also worth noting that this map structure requires that there are multiple easy to find landmarks present in the environment such as pillars.[22]

Google Cartographer is an open-source system for SLAM. Google Cartographer can be configured to use different types of sensors, however, for the purpose of this work, the implementation using a LiDAR sensor is of interest. Also note that while the LiDAR sensor provides the measurements of the environment, odometry data can also be supplied to the implementation in order to further improve the performance of the SLAM process. Google cartographer uses the occupancy grid map structure described in Section 2.3 with a predefined cell size of 5×5 cm, this value can however be changed if desired.

2.3 Occupancy Grid

In order to be able to use a map, the map must have a defined structure. There are many ways that a map can be structured to represent the true environment. However, since the goal of the work is to be able to not only describe but also use the features in the environment belonging to semi-static objects, the map structure must be able to describe these objects. For a 2D environment, an occupancy grid is a common representation that is not limited to only describing linear or straight objects such as walls. The occupancy grid consists of a grid of individual cells of the environment where each cell represents the probability of the area being occupied in the true environment. A common structure for a 2D environment is to use squared cells in a 2D array to represent the environment. Since a map of this structure has a limited resolution, the cells do not only contain a binary value indicating that the cell is occupied or empty but instead consists of a value in a predefined interval giving a probability measurement of the cell being occupied or empty. A visualization of the structure can be seen in Figure 2.2.

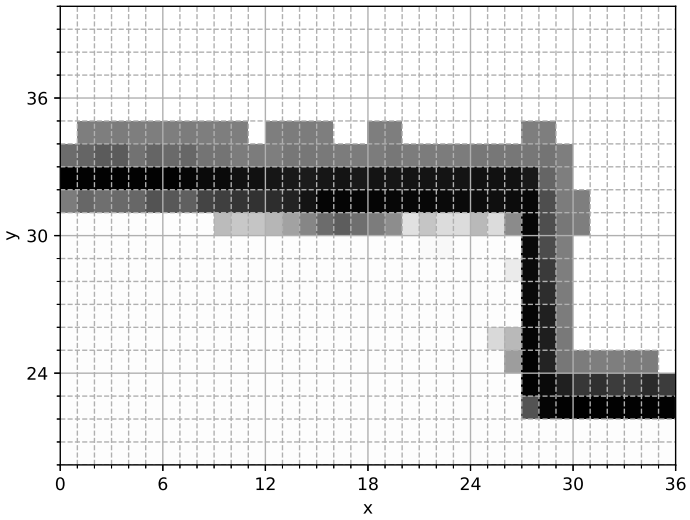


Figure 2.2: Example of an occupancy grid map of three connected wall segments. Darker color indicates a higher probability of the cell being occupied.

In Figure 2.2, three connected wall segments are represented using the occupancy grid map structure, due to measurement noise the map does not contain clear walls but instead approximates the distribution of the measurements made using the non-binary values for each cell. From Figure 2.2 the assumption can be made that a smaller cell size will enable a more detailed description of the objects in the true environment and vise versa.

2.3.1 Finding the corresponding cell

In order to be able to use the map, there must be a clear model that is able to take a measured point from the real world and correlate it to a cell on the map. Since the map in constructed as a grid, the index of the grid cell corresponding to the coordinate must be found. In order to do this the size of each cell must be taken into consideration. Given that there is no offset needed such that all possible measurements correspond to a positive map coordinator the corresponding cell can be found using as follows. Given a measurement z_i corresponding to the map coordinate $(x, y)^T$. The corresponding cell is

$$(x_{index}, y_{index})^T = floor\left((x, y)^T \frac{1}{res}\right), \tag{2.1}$$

where res is the resolution of the map, for example, $res = 0.05$ for a map consisting of 5×5 cm cells and a map scaled in meters. The floor operator ensures that the index is truly an integer index meaning that, as mentioned, multiple measurements can fall into the same cell even if they are slightly different. If the occupancy grid is supposed to represent an area on the map that does not have its origin in zero and only uses positive coordinates, an offset must be used. The offset ensures that

all measurements of interest yields a positive index and it can also ensure that the occupancy grid is not unnecessarily large. Assuming offset x_{offset}, y_{offset}, the corresponding cell index is found using:

$$(x_{index}, y_{index})^T = floor\left((x - x_{offset}, y - y_{offset})^T \frac{1}{res}\right) \tag{2.2}$$

2.3.2 Value Function

When using an occupancy grid map structure, the optimal state of the AGV given the measurements is difficult to obtain in a feasible way. Instead, the map is used to give a value of how good or bad the current estimation of the state is, a so called fitness value. The value function is used to derive the fitness value. While the value function can be structured in different ways, given that the measurements are taken using a LiDAR sensor, a simple way of structuring the value function is to simply use the sum of all cell values for all measurements. This approach for deriving the fitness value based on a observation, a pose estimate and a map was implemented in [9] as

$$F(z) = \sum_{i=0}^{n} m(x_{index,i}, y_{index,i}) \tag{2.3}$$

where F is the fitness for the state of the robot given an observation z with n measurements of cells with occupancy value $m(x_{index,i}, y_{index,i})$ corresponding to the index $[x_{index,i}, y_{index,i}]$ based on measured coordinates x_i and y_i as presented in (2.2).

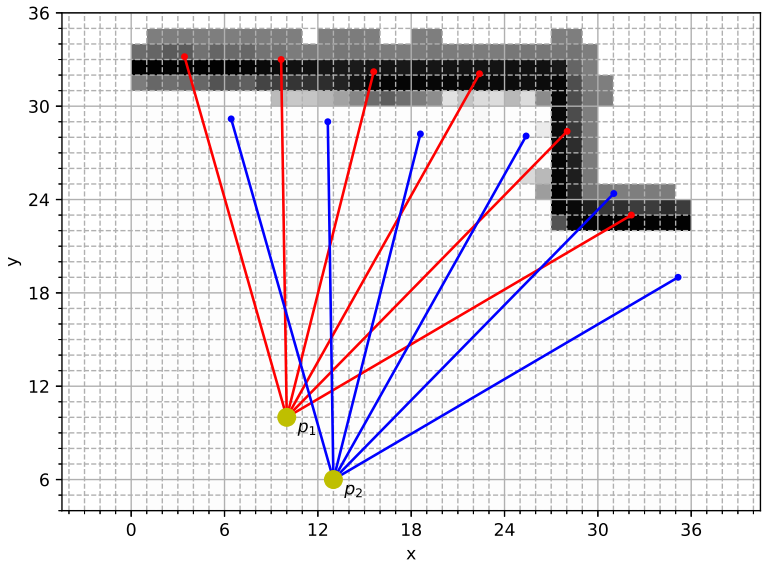


Figure 2.3: Example of an occupancy grid map used for evaluating two estimated states using an observation, the red and blue lines are the rays of the LiDAR sensor and the distance measured.

Using cell values ranging from 0-100 (white = 0, black = 100) and a given measurement, Figure 2.3 showcases an example where vehicle state p_1 gives a larger fitness

value (454) compared to p_2 (50), which can later on be used in order to localize in the environment. The fitness values for p_1 and p_2 are derived using the value function shown in (2.3).

2.4 Normal Distributions Transform

Normal Distributions Transform is a type of data representation first described in [10]. The NDT map representation is similar to the occupancy grid representation with some key differences. In the NDT representation, the cells indicate if a cell is occupied or not but instead how the data inside of the local cell is distributed. The distribution of the data is estimated using a normal distribution, thereby the name Normal Distributions Transform. If no measurement has been taken of a cell, the cell is left as empty meaning that only cells that have been measured will contain a normal distribution.

2.4.1 Cell

The cells in the NDT structure are defined using normal distributions describing the data measured in the area that the cell is representing in the real world. This means that the cell can indicate the shape of the data and thereby the shape of the objects in the real environment. The sample mean, μ_j, and covariance, Σ_j, of the cell j is derived using n number of measured points $[x_i, y_i]^T$ corresponding to the area belonging to the cell as

$$\begin{aligned} \mu_j =& \frac{1}{n}\sum_{i=1}^{n}\begin{bmatrix} x_i \\ y_i \end{bmatrix} \\ \Sigma_j =& \frac{1}{n-1}\sum_{i=1}^{n}\left(\begin{bmatrix} x_i \\ y_i \end{bmatrix} - \mu\right)\left(\begin{bmatrix} x_i \\ y_i \end{bmatrix} - \mu\right)^T. \end{aligned} \tag{2.4}$$

An example of how an NDT map would be constructed using data points measured at a corner between two walls can be seen in Figure 2.4.

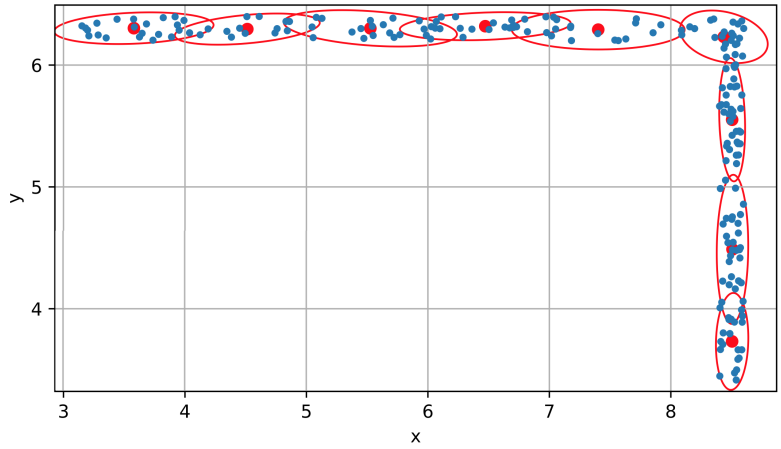


Figure 2.4: Example of a derived NDT map where measurements indicated by blue dots are used to derive the distributions inside of each cell corresponding to a 1×1 distance unit cell on the map. The normal distributions are derived using (2.4) for each individual cell where the red dot represents the mean of the cell and the red ellipse represents the 2-σ level curve of the covariance. Note that the scale of the map is arbitrary since the actual size of a cell can be set to different values.

To further illustrate the capability of the NDT cell, a segment of a circular object could for example be represented in the map as can be seen in Figure 2.5.

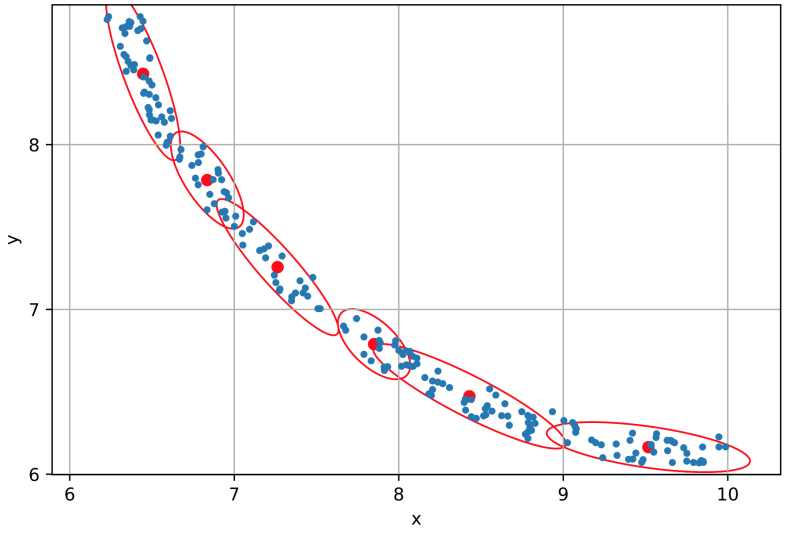


Figure 2.5: Example of a derived NDT map based on measurements taken on a circular object. The objects are represented in the same way as for Figure 2.4.

2.4.2 Value Function

When localizing using the NDT map, a way to actually use the map must be derived. Different types of filters allow for different types of information gained from using the data. In [10] and [12] the normal distributions in each cell were used to derive a

score or fitness value for a given observation and pose. The suggested method was to look at the cell corresponding to a measurement, as described in Section 2.3.1, and calculate the likelihood of the measurement given the pose hypothesis by analyzing the distribution in the NDT cell on the map. The contribution to the likelihood of a single measurement z_i in an observation $\mathbf{z}$ and for a pose $\mathbf{p}$ given m cells being present in the map is

$$\mathbf{p}(z_i) \propto \sum_{j=0}^{m} \exp\Big(- \frac{(z_i - \mu_j)^t \Sigma_j^{-1} (z_i - \mu_j)}{2} \Big), \tag{2.5}$$

In [10] and [12] it was suggested that given a current position estimate of the sensor $\mathbf{p}$, the fitness of the estimation should be derived as

$$F = \sum_{j=0}^{m} \sum_{i=1}^{n} \exp\Big(- \frac{(z_i - \mu_j)^t \Sigma_j^{-1} (z_i - \mu_j)}{2} \Big), \tag{2.6}$$

where z_i is the x_i and y_i coordinate of the transformed measurement given the position of the sensor $\mathbf{p}$ with a total of n measurements. And μ_j, Σ_j being the mean and covariance corresponding to the cell j. Note again that if a cell has never been measured before, the contribution will be zero since there will not be a normal distribution present in that cell.

It was however concluded in [10] and [12] that deriving the likelihood contribution for all measurements given all cells on the map is not a feasible approach based on time complexity. Instead only the cell corresponding to measurement is used to estimate the likelihood of the pose given a observation as

$$F = \sum_{i=1}^{n} \exp\Big(- \frac{(z_i - \mu_j)^t \Sigma_j^{-1} (z_i - \mu_j)}{2} \Big), \tag{2.7}$$

where cell j corresponds to the measurement z_i

2.5 Occupancy Grid to NDT Conversion

Converting a map from the occupancy grid map structure the NDT structure would enable a system to be created using a pre-existing SLAM implementation based on the occupancy grid representation while still using the NDT structure when localizing. This concept was looked at further in [13]. In the conversion, they chose to increase the size of the NDT cells to represent 30×30 cm in the real environment. Comparing to a typical cell size of 5×5 cm [7] means that 36 cells will now be converted into one. An important note is, therefore, to mention that the occupancy cell only represents a probability of the cell being occupied or empty while the NDT cell is able to describe the shape of the object inside of the cell, as was visualized partly in Figure 2.4 and even more clearly in Figure 2.5 where a circular object was tested. The basic algorithm proposed in [13] is presented in Algorithm 1.

Algorithm 1 Pseudo code for map conversion algorithm.

1: **for all** $ndtCell_{x,y}$ in $ndtGrid$ **do**
2: $occupancyGridCells :=$ `getAssociatedCells`$(ndtCells_{x,y})$
3: Initialize empty set C
4: **for all** occupied $occcell_{x',y'}$ in $occupancyGridCells$ **do**
5: add $occcell_{x',y'}$ with weight w to set C, where w is the occupancy probability of cell $occcell_{x',y'}$
6: **end for**
7: $mean_{x,y} :=$ `calculateWeightedMean`(C)
8: $covariance_{x,y} :=$ `calculateWeightedCovariance`$(C, mean_{x,y})$
9: **end for**

In Algorithm 1 the `getAssociatedCells` method collects the chosen number of cells corresponding to the area that the new NDT cell represents in the environment. The `calculateWeightedMean` and `calculateWeightedCovariance` methods are derived as weighted multivariate mean and sample covariance as

$$\begin{aligned} \mu &= \frac{1}{\sum_{i=1}^{n} w_i} \sum_{i=1}^{n} w_i \begin{bmatrix} x_i \\ y_i \end{bmatrix} \\ \Sigma &= \frac{1}{\sum_{i=1}^{n} w_i} \sum_{i=1}^{n} w_i \left(\begin{bmatrix} x_i \\ y_i \end{bmatrix} - \mu \right) \left(\begin{bmatrix} x_i \\ y_i \end{bmatrix} - \mu \right)^T, \end{aligned} \tag{2.8}$$

where w_i is the value of the occupancy grid cell. The condition that an occupancy grid cell is to be considered occupied was in [13] set to the value of the cell being at least 55 on a scale from 0-100. To improve the performance of the algorithm, not only the center of a cell was added as a point but also the corners of that cell, providing that the corner was not already present in the set C. To illustrate this, the same wall segments that are present in Figure 2.4 will be used with the addition of the points present in the set C. The result can be seen in Figure 2.6. Note that the reason why not all present occupancy grid cells have points generated at them is because of the criterion that the value of a cell must be at least 55 to be considered occupied. Also, note that the scale of the cells are set so that an occupancy grid cell has the size 1×1 giving the NDT cell a size of 6×6, this was done to improve the readability of the figure since the chosen size of the two cell structures can be changed regardless of the values chosen in [13].

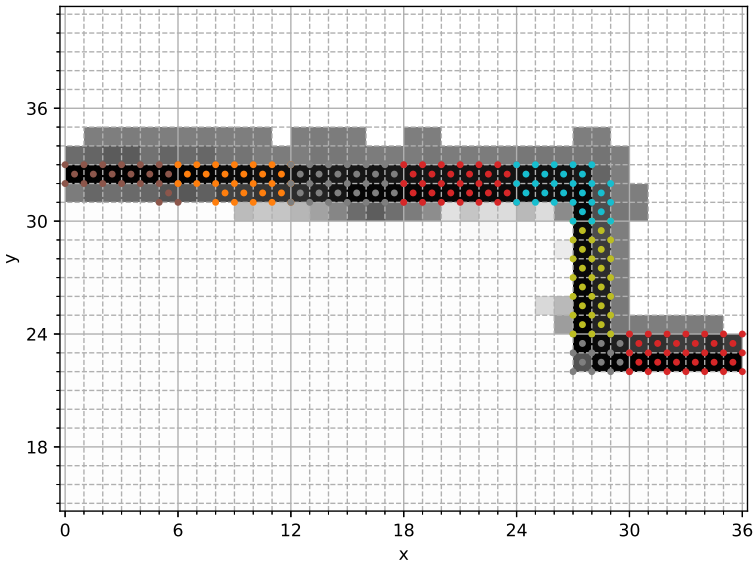


Figure 2.6: Points present in the set C for each NDT cell where each color represents the point in the set C for a single NDT cell. The image indicates that not only the center but also the corners of a cell is added to the set C. Cells having a too low occupancy probability are not allowed to add any points to set C.

The mean and covariance are then derived using the set C for each NDT cell with weights w_i corresponding to point $(x_i, y_i)^T \in C$ using (2.8). The derived NDT cell is replacing a 6×6 grid of occupancy grid cells. The resulting map conversion can be seen in Figure 2.7.

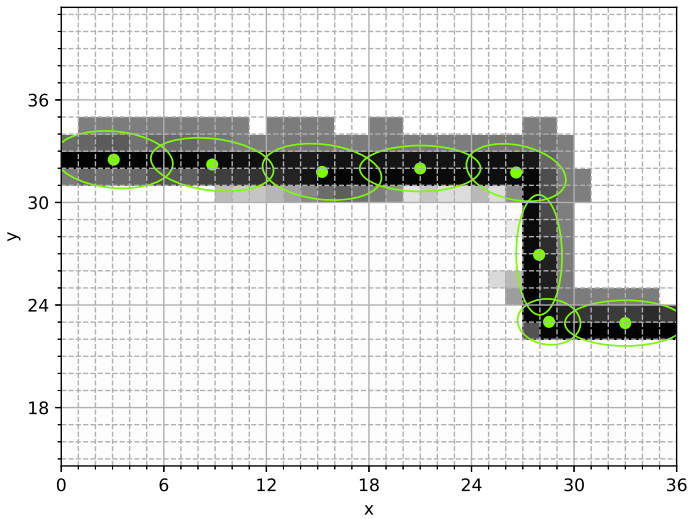


Figure 2.7: Resulting NDT map generated from points shown in Figure 2.6 and using mean and covariance described in (2.8)

2.6 Particle Filter

Measurements usually have some measurements noise and the noise can yield a bad estimate of the specific state the measurements provide. To give a more robust and more accurate estimate of a certain state some type of filter is used. One collection of filters is the Bayesian filters, these filters tries to estimate the probability density function of the estimated state using the measurements in sequence. These Bayesian filters only depend on the previous estimated probability density function and the current estimate, making it a Markov Chain process. One type of Bayesian filter is the particle filter that is commonly used to solve non-linearities. Particle filter is used to estimate the posterior probability density function of a certain state $\mathbf{x}_t$ given some measurements z_t where t is the current time instance. One objective of the work is to give the best possible estimate of the state, i.e. position and heading, $\mathbf{x} = [x, y, \theta]^T$. The estimate of the posterior probability density function of the state is $p(\mathbf{x}_t|\mathbf{z}_t)$, where $\mathbf{z}_t = \{\mathbf{z}_1, ..., \mathbf{z}_t\}$ is the measurement sequence [23]. The posterior probability density function of the state is given by

$$p(\mathbf{x}_t|\mathbf{z}_t) = p(\mathbf{x}_0) \prod_{i=1}^{t} p(\mathbf{z}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{x}_{t-1}), \tag{2.9}$$

where $p(\mathbf{z}_t|\mathbf{x}_t)$ is the likelihood of $\mathbf{x}_t$ given the measurement sequence $\mathbf{z_t}$, $p(\mathbf{x}_0)$ is the prior probability density function of $\mathbf{x}$, and, $p(\mathbf{x}_t|\mathbf{x}t-1)$ is the prediction of $\mathbf{x}_t$ given the previous $\mathbf{x}_{t-1}$.

The basic idea of how a particle filter calculates (2.9) given all observations up to the current time is that it uses many random hypothesizes, called particles. Each particle has a weight which describes the probability that the state of the particle is the true state. Using these particles, the probability function can be described as

$$p(\mathbf{x}_t|\mathbf{z}_t) = \sum_{i=1}^{N} w_t^i \delta(\mathbf{x}_t - \mathbf{x}_t^i), \tag{2.10}$$

where w_t^i is the weight and $\mathbf{x}_t^i$ of particle i at time t. δ is a Dirac impulse function and N is the number of particles.

The particles are propagated using the prediction model, which estimates the new state of the particles using the prediction model with an added noise. Another alternative is to use measurements from sensors that measure the movement of the AGV such as wheel encoders. In order to account for any error in the model the state of each particle are sampled with the motion model, $\mathbf{x}_t^i \sim p(\mathbf{x}_t|\mathbf{x}_{t-1}^i)$, where $\mathbf{x}_t^i$ is the state of the particle i at time t.

When the particles have been propagated, the particles are evaluate with the likelihood of how well the particle match with the measurement. The likelihood of each particle is then used to update the weight of the specific particle according to

$$w_t^i \propto w_{t-1}^i p(y_t|\mathbf{x}_t^i) \tag{2.11}$$

where $p(y_t|\mathbf{x}_t^i)$ is the likelihood of the ith particle. The weights are then normalized with the sum of all weights of the particles.

2.6.1 Resampling

As the particles are propagated, the spread of the particles will increase. In order to prioritize particles with higher weights, resampling is performed. The resampling step takes the weight and state of all particles and redistributes particles according to their weight values. This ensures that multiple particles will be in states with high weight prior to the next step of propagation [23]. A common resampling method is to use residual resampling [24]. The residual resampling distributes the particles to states evenly according to their weight such that the number of particles in a state post resampling is proportional to the weight of the state $p(\mathbf{x}_t) \propto w_t^i / \sum_{k=0}^{n} w_t^k$ where n is the number of particles [25]. Since the time complexity of the resampling is not negligible it is possible to not resample at every iteration of the particle filter and instead resample once a criterion is met. A common choice of criteria is the covariance of the states of all of the particles, meaning that the resampling is only performed once the particles have diverged to a certain covariance.

2.7 Dual-Time NDT

To increase the robustness when the environment is changing a dual timescale map (DT map) was evaluated in [15]. Dual-Time NDT is a small extension of the previous described NDT. The key difference is that instead of just have one static map the Dual-Time NDT has a second map that describes the current state of the environment with all the objects including the static- and dynamic objects. The authors used the Monte-Carlo localization also referred to as a particle filter as described in Section 2.6. Dual-Time NDT uses a map describing the current state of the environment with the static and semi-static objects and not only the static objects. The DT map consists of both the static map and the dynamic map. The dynamic part of the map helps increase the robustness in the estimation due to the environment shifting over time as objects are moved around and parts of the static map are obscured by the dynamic object being placed between the static objects and the LiDAR. A key difference between the formulation of the static and dynamic map is that an occupancy probability was added to the cells in the dynamic NDT map described in (3.13). Similar to the cells in an occupancy grid map, the occupancy probability is supposed to denote the probability of a cell being occupied. This value is later on used during localization to weight measurements measuring cells will higher occupancy probability higher than cells with a lower occupancy probability.

The pseudo code for the use of a dual-timescale map suggested in [15] can be seen in Algorithm 2.

Algorithm 2 Dual-Timescale NDT-MCL psuedo code

```
propagateParticles
Build a local NDT map from the observations
for all Particles x_k^i do
    F = 0
    Transform local NDT map according to p_i
    for all NDT_i in local map, i = 1...N do
        f_i = l(c_zi, c_m_s i) as in (2.14)
        if f_i < ξ then
            f_i = l(c_zi, c_m_t i) p(c_i = 1) as in (2.14)
        end if
    end for
    F = F + f_i
end for
normalizeWeights
resample
estimatePose, mean of all particle poses
Σ_trace = trace(Σ_position), sum of diagonal elements of the particle pose covariance
if Σ_trace < γ then
    Update Dynamic map
end if
```

In [10, 12] an evaluation function for NDT was described, in the newer publication [15] a different way of evaluating the current estimate was developed, in this case the particles pose. In [15] it was chosen to generate a new NDT map $\bar{\mathbf{z}}_t$ given the measurements for a observation (scan) $\mathbf{z}_t$ in a local coordinate frame. For each particle, each found distribution in the new NDT map $\bar{\mathbf{z}}_t$ was then rotated and translated to the perspective of the current particle position $\mathbf{x}_t$ and compared to the corresponding cell in the reference map denoted m. Since the NDT map is only an approximation of the environment it was also suggested to not only look at the corresponding map cell but also the neighboring cells to search for a distribution matching the measurement, this method will from here on will be denoted as using a search space. The fitness contribution of a given observed NDT cell $c_{zi} \in \bar{\mathbf{z}}_t$ with mean and covariance μ_{zi}, Σ_{zi} and a reference cell $c_{mi} \in m$ with mean and covariance μ_{mi}, Σ_{mi} is then derived as

$$f_i = l(c_{zi}, c_{mi}) = \exp\left(- \frac{(\mu^{'}{}_{zi} - \mu_{mi})^t (\Sigma^{'}{}_{zi} + \Sigma_{mi})^{-1} (\mu^{'}{}_{zi} - \mu_{mi})}{2} \right) \tag{2.12}$$

where f_i is the fitness of particle i, and is used in Algorithm 2. The total fitness for a given particle using only one map is

$$F = \sum_{i=1}^{N_{zt}} f_i = \sum_{i=1}^{N_{zt}} l(c_{zi}, c_{mi}) \tag{2.13}$$

with N_{zt} being the number of distributions found in $\bar{z}_t$ and F is the total fitness as in Algorithm 2.

Using both of the two maps, the fitness evaluation is changed to also take into account the dynamic map. The static map will now be denoted as m_s and the dynamic map as m_t and the value function instead becomes

$$F(\bar{\mathbf{z}}_t|\mathbf{x}_t, m) = \sum_{i=1}^{N_{zt}} \begin{cases} l(c_{zi}, c_{msi}) & , \text{ if } l(c_{zi}, c_{msi}) > \xi \\ l(c_{zi}, c_{mti})p(c_i = 1) & , \text{ otherwise} \end{cases} \tag{2.14}$$

With $p(c_i = 1)$ corresponding to the occupational probability of the cell c_{zi} and ξ being used to give the static map priority over the dynamic map while enable the use of the dynamic map whenever the static map finds a matching distribution that does not give a high enough probability value.

2.7.1 Map Update

From Algorithm 2 it can be seen that the map is updated after each iteration of the particle filter given that the trace of the covariance of the particles is below a threshold γ. In the map update step, the dynamic map is updated such that both the occupancy probability as well as the mean and covariance of each cell in the dynamic NDT map is updated according to [16]. To derive the mean and covariance according to (2.8) all of the points need to be stored for all different observations. This is infeasible since the amount of data will increase linearly over time meaning that both the time- and memory complexity will grow over time. This problem was solved in [16] where a method of precisely deriving the new mean and covariance by only storing the number of measurements included in the current mean and covariance. Similarly to the value function l it was in [15] and [16] proposed that the map update should not be updated using single measurements but instead derive an NDT map for the current observation and merge the current dynamic NDT map and the NDT map generated from the current observation. Following [16], given two cells c_1, c_2 with mean μ_1, μ_2, sample covariance Σ_1, Σ_2 with n_1, n_2 number of points used to generate the cells, respectively, the combined mean $\mu_{1\oplus 2}$ of the two cells should be derived as

$$\mu_{1\oplus 2} = \frac{1}{n_1 + n_2}(n_1\mu_1 + n_2\mu_2) \tag{2.15}$$

And the combined sample covariance $\Sigma_{1\oplus 2}$ as:

$$\Sigma_{1\oplus 2} = \frac{\left(\Sigma_1(n_1 - 1) + \Sigma_2(n_2 - 1) + \frac{n_1 n_2}{n1+n_2}(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T\right)}{n_1 + n_2 - 1} \tag{2.16}$$

To update the occupancy probability $p(c_i = 1)$ for a given cell c_{zi}, it is according to the authors in [16] not enough to analyze the cell measured by the observation. The cells that are not measured must also be taking into account since it is important to know if there is not an object in a cell. This is in [16] performed by ray-tracing from the origin of the sensor to the measured cell. All cells that lie in between these points must have not been hit. However, since the premise of using the NDT structure is that the object in a given cell is not homogeneous, a cell can therefore be missed and measured by two separate measurements for a given observation. In

[16] it was therefore suggested that a cell should be considered as being occupied if a single measurement is taken inside the cell even if multiple ray-tracing indicates that the cell is empty for a given observation.

To get an occupancy probability in the range $[0, 1]$ the update step instead changes the log-odds p_{log} for each cell. Since the mapping from log-odds to probability takes log-odds values in the range of $[-\infty, \infty]$ and maps to a probability value in the range $[0, 1]$ the log-odds value is a suitable choice. When updating the log-odds value for each cell, the authors of [16] suggested the following update equation for a single cell c_i given observation z_t, where N_p are the number of points measuring inside the cell and N_e are the amount of measurements that did not fall inside the cell even though the ray-tracing indicates that the cell would have been measured if it would have been homogeneously occupied:

$$p_{log}(c_i = 1|z_{1:t}) = p_{log}(c_i = 1|z_{1:t-1}) + \begin{cases} N_p log(\frac{p_0}{1-p_0}), & \text{if } N_p > 0 \\ N_e log(\frac{p_e}{1-p_e}), & \text{if } N_p = 0 \end{cases} \tag{2.17}$$

In [16] the suggested tuning parameters are $p_0 = 0.6$ and $p_e = 0.49$.

In later articles, problems using the proposed (2.15) and (2.16) were brought up. Namely that if the object inside of a cell is moved, using the true covariance for all measurement is undesirable since the true content of the cell in the real environment does not depend on how the content inside a cell where distributed in a previous time-instance if the object has moved. In [17] and [18] it was therefore suggested to use a regency weighted recursive sample covariance and mean update. It was suggested to use the number of points N_p for estimating the mean and covariance, which compared to (2.15) and (2.16) are instead updated as:

$$\mu_{1\oplus 2} = \frac{1}{N_p + n_2}(N_p\mu_1 + n_2\mu_2) \tag{2.18}$$

$$\Sigma_{1\oplus 2} = \frac{\left(\Sigma_1(N_p - 1) + \Sigma_2(n_2 - 1) + \frac{N_p n_2}{N_p + n_2}(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T\right)}{N_p + n_2 - 1} \tag{2.19}$$

Where the value of N_p is update after each update of the cell distribution according to:

$$N_p = \begin{cases} n_1 + n_2, & \text{if } n_1 + n_2 < M \\ M, & \text{if } n_1 + n_2 \geq M \end{cases} \tag{2.20}$$

Where M is a tuning parameter. The effect of this is that a lower value of M gives a faster adaptation to the changes made in the cell while a larger value of M ensures that the information gained about the cell has a larger effect on the mean and covariance for a longer period of time.

3

Methods

The main problem can be divided into two parts, map the environment to create a static map and localize using the created map as can be seen in Figure 3.1. The localization also depends on which motion model and measurement model that is used. In this Section the motion model and measurement model together with the coordinate systems will be presented. A probable explanation why the NDT map yields a more accurate estimate than the occupancy grid map and therefore is used is presented. After that the two main parts is presented starting with the static map creation followed by the localization using the map. While the division into two parts is a rather simplified description, a more detailed block diagram can be seen in Figure 3.1. The underlying prerequisites and theory brought up in Section 2 will here be used and modified to suit the problem at hand. This Section will also propose a way to update the static map over time to become more and more accurate in regards to only containing static objects, this will be done using the dynamic map generated online as the AGV navigates in the environment. This procedure can also be seen in Figure 3.1.

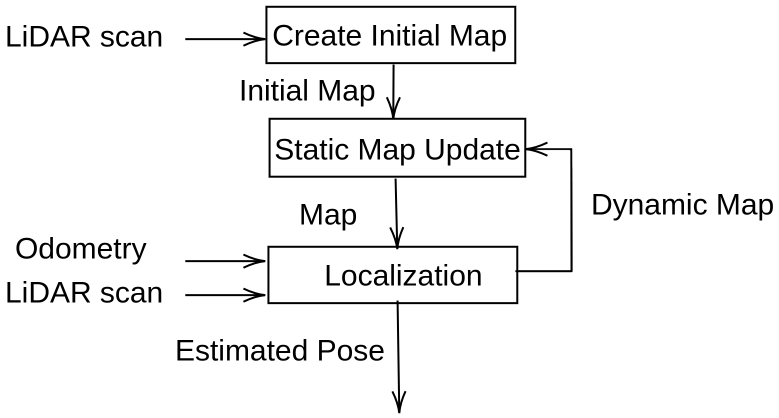


Figure 3.1: Block diagram over the system.

3.1 Motion Model/Odometry

The chosen state vector contains the x and y coordinate of the AGV as well as the heading, θ. Since the floor of the environment is assumed to be level at all times,

the height, and tilt of AGV is not of interest. In order for the localization to be as accurate as possible, the motion of the AGV will be used to predict the new state of the particles.

The AGV has two wheel encoders. Wheel encoders give an incremental change in movement which means that the sensors do not give an absolute position and only give a change in position. The encoders have a resolution of how many pulses per revolution (PPR). In every update, the encoder gives how many pulses the wheel has traveled in one time step. Then the distance the wheel has traveled in one time step can be determined as in (3.1)

$$\Delta l = \frac{2\pi r \cdot pulses}{PPR} \tag{3.1}$$

Where r is the radius of the wheel. This is done for both wheels since the AGV has a differential drive system, this means that the AGV has two motors that drive two wheels independently of each other. To change the direction of the vehicle the motors run at different speeds.

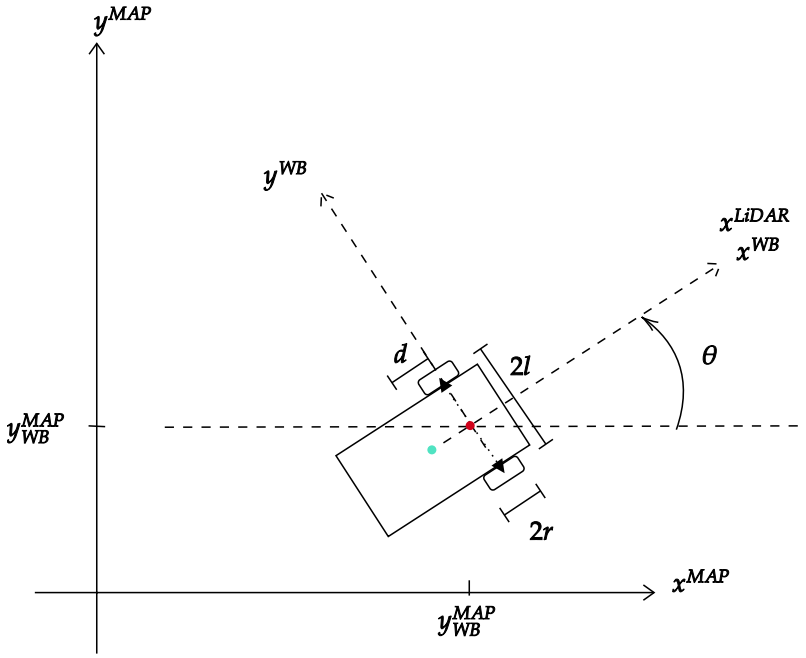


Figure 3.2: Differential drive where the cyan dot is the LiDAR and the red dot is the wheel base.

The positional difference is in the coordinate system of the wheelbase which has an offset from the LiDAR sensor as can be seen in Figure 3.2. It is important to have the position of the wheelbase in the global map coordinate system (y_{WB}, y_{WB}). Given the previous position and rotation, $(x_{WB_{t-1}}, y_{WB_{t-1}}, \theta_{WB_{t-1}})$, of the wheelbase in the map frame and a change in left and right wheel the transformation is done by a linearized model which requires the arch of the trajectory to be short. The update frequency for the odometry data is 42 Hz and the linearization is therefore

considered as a valid approach. The equations describing the motion model can be found in (3.2).

$$\begin{aligned} \Delta\theta &= \frac{\Delta r - \Delta l}{2l} \\ \Delta x &= \frac{(\Delta l + \Delta r) \cdot \cos(\theta + \Delta\theta/2)}{2} \\ \Delta y &= \frac{(\Delta l + \Delta r) \cdot \sin(\theta + \Delta\theta/2)}{2} \end{aligned} \tag{3.2}$$

where Δr and Δl is the difference in position for the right and left wheel. Using (3.2), the new position of the AGV can be found using (3.3).

$$(x_{WB_t}, y_{WB_t}, \theta_{WB_t}) = (x_{WB_{t-1}} + \Delta x, y_{WB_{t-1}} + \Delta y, \theta_{WB_{t-1}} + \Delta\theta) \tag{3.3}$$

A noise will also be added to the motion model as presented in Section 2.6. How the noise was added will be explained in a more detailed manner in Section 3.5.1.

3.2 Measurement Model

The measurement model contains two parts. The first part is to take a measurement and gives the measurement in the coordinate system of the AGV. The second part is to transform the measurement to the coordinate system of the map.

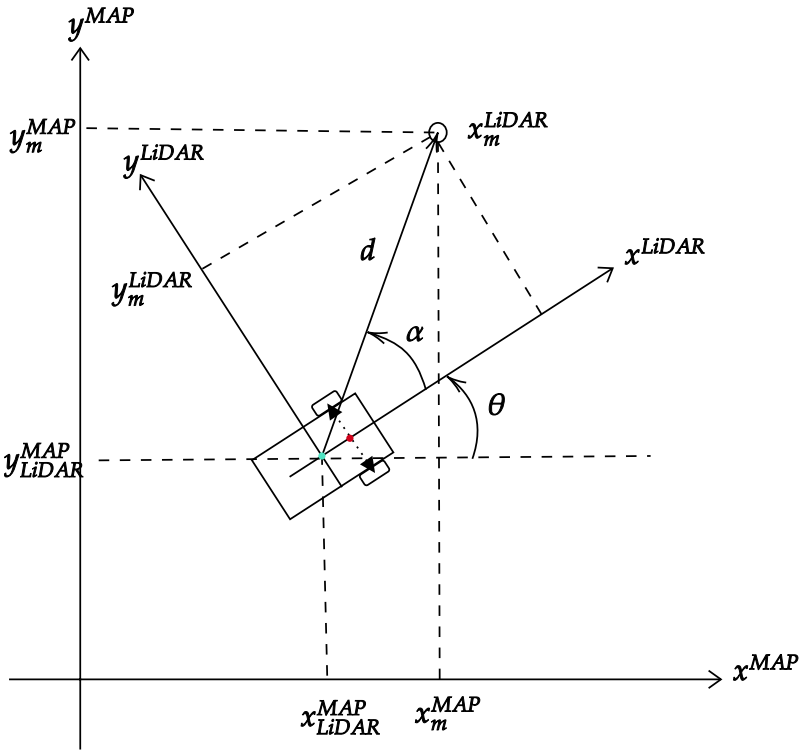


Figure 3.3: Coordinate system with a measurement

From the perspective of the AGV, the measurement model of the LiDAR is a range bearing model as can be seen in Figure 3.3. This means that the sensor gives for

each laser measurement a distance, d, and an angle, α, to the object the laser beam hits. To get the measurement point in the AGV coordinate system as in (3.4) is used.

$$\begin{bmatrix} x_m^{AGV} \\ y_m^{AGV} \end{bmatrix} = \begin{bmatrix} d \cdot \cos\alpha \\ d \cdot \sin\alpha \end{bmatrix} \tag{3.4}$$

However to be able to match the measurements with the map the measurement needs to be in the map coordinate system. Given the position and angle of the sensor in the map frame and a measurement from the sensor the measurement point in the map frame is calculated in (3.5)

$$\begin{bmatrix} x_m \\ y_m \end{bmatrix} = \begin{bmatrix} d \cdot \cos(\alpha + \theta) + x_{AGV} \\ d \cdot \sin(\alpha + \theta) + y_{AGV} \end{bmatrix} \tag{3.5}$$

3.3 Normal Distributions Transforms vs Occupancy Grid

While the optimal way to estimate the distribution of an NDT cell would be to have access to an infinite amount of measurements inside of the cell, saving all of the data during the initial mapping is to be considered infeasible for the specific case of this work. The conversion presented in [13] is a good compromise between precision and computational efficiency. While some of the knowledge of the environment could be lost during the conversion it is important to note a difference between the occupancy grid cell and an NDT cell. Namely that the NDT cell has an optimum in its value function contribution, described in Section 2.4.2 compared to the contribution from the occupancy cells described in Section 2.3.2. To showcase this difference, a 1D case is first analyzed.

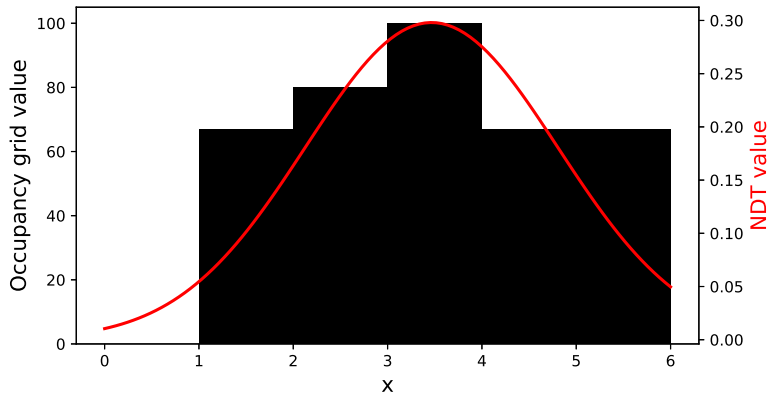


Figure 3.4: A 1D demonstration of a cell that in both occupancy grid and NDT representation.

Looking at Figure 3.4, the conclusion can be drawn that while the NDT cell might not represent the true environment in a more precise manner, it does have a maxima

and therefore an optimal value of x to get the largest fitness value. The same can not be said for the occupancy grid cells since all values of $x \in [3, 4]$ give the same fitness value. While the effect of this might be mitigated as the number of measurements are increased from one as in Figure 3.4 to a larger number the same phenomenon can be found using the intended 2D representation and more points.

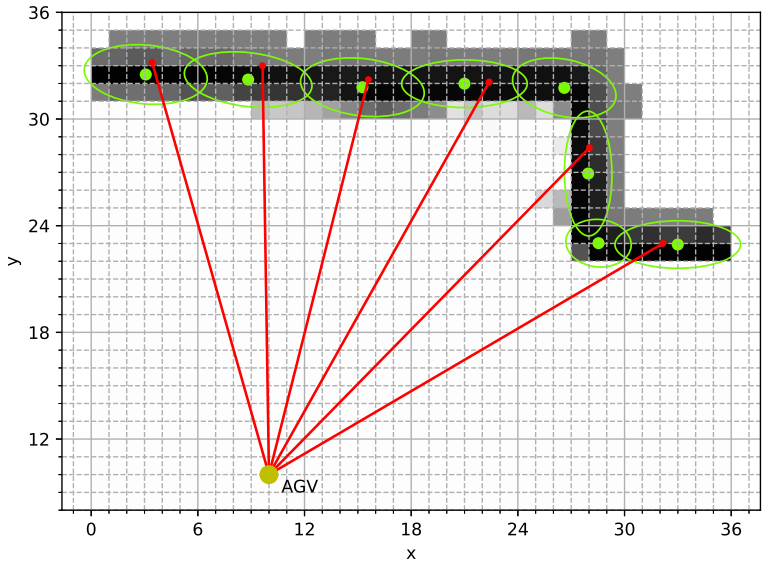
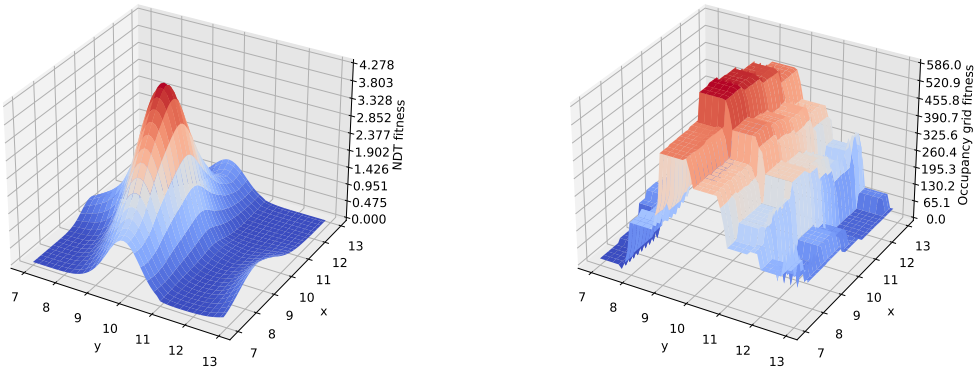


Figure 3.5: A 2D demonstration of a cell that in both occupancy grid and NDT representation.

In Figure 3.5 a state of the AGV is given and a few measurements are used to derive a fitness value based on both an occupancy grid map and an NDT map converted using the method proposed in [13]. By allowing the AGV to move slightly in the x and y direction while keeping the direction constant, the fitness value based on the position of the AGV can be derived. The fitness value can be illustrated as a surface based on the position of the AGV and can be seen in Figure 3.6.



(a) Using the NDT value function

(b) Using the occupancy grid value function

Figure 3.6: Fitness values of the AGV seen in Figure 3.5 while allowing the AGV to change x and y position

The NDT map provided an optimal position of the AGV while the occupancy grid map only provides a region of optimally meaning that the AGV will not be able to converge to the same position each time. The reason for choosing to implement a localization based on the NDT map structure is based results seen in [6, 13, 14, 15], however, the concept shown here is potentially the cause of this difference in accuracy and more importantly repeatability.

3.4 Initial Map Creation

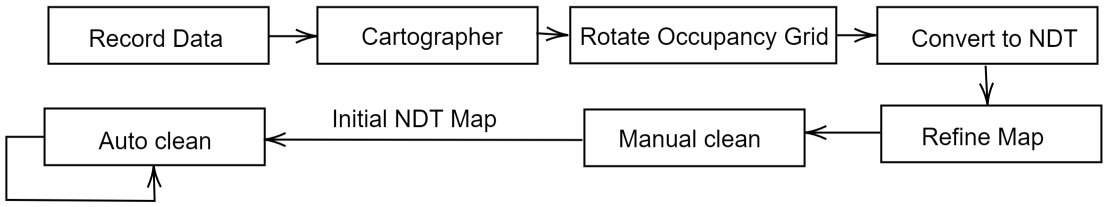


Figure 3.7: Block diagram over how the initial map is built.

To be able to localize in the environment an initial map has to be constructed. This is done in several steps. In Figure 3.7 an overview of the workflow for building an initial map is visualized. The different blocks will be presented in more detail.

3.4.1 Record Data

The LiDAR sensor is connected to the laptop via the ethernet port. A ROS node reads the data from the LiDAR sensor continuously. The ROS node published the data on a topic so that other nodes can read the data. A similar node reads the data regarding the wheel odometry using the serial port and published the data on a separate topic. While the nodes are running, the AGV is manually driven by an operator throughout the environment such that the LiDAR has seen all the walls, pillars, corners, etc. While moving the AGV around the LiDAR and odometry topics are recorded and saved as a rosbag. The rosbag containing the data from the LiDAR sensor and the odometry is later used offline to build the initial static map.

3.4.2 Generate Occupancy Grid using Google Cartographer

To build an initial map Google Cartographer was used. Cartographer uses the LiDAR data and builds an occupancy grid from no prior knowledge of the environment. The data that is used by Cartographer is the pre-recorded data from the LiDAR when it was moving around in the environment. The occupancy grid cell size that was chosen but not evaluated was the default size 5×5 cm.

The mapping is only done with the LiDAR and not the odometry even if the odometry could potentially improve the map. This is because more information about the AGV could give a better estimate of the position and therefore a better map. The reason for not using the odometry data is based on a suggestion from the company. In the future, to map an environment without bringing an entire AGV is preferable. This way the company can perform the mapping before they build the AGV and the planning of routes of the AGV can be performed before the installation of the AGV fleet. The possibility to use data with only a LiDAR at the correct height could shorten the time from constructing the AGV's to a fully running fleet which could save both money and time.

3.4.3 Rotate Occupancy Grid

The occupancy grid built by Google Cartographer is based on the initial state of the AGV. The initial heading of the AGV becomes straight in the x-axis of the coordinate system of the map. This may be undesirable and the map needs to be rotated and shifted to the wanted coordinate system. The occupancy grid can be seen as an image, with one layer, where the cells in the occupancy grid are the pixels in the image. The grid can be rotated with an angle and then shifted. Normal rotation of images could lead to lost information and this is solved by increasing the size of the grid such that all the cells in the original grid will still be in the new rotated grid.

3.4.4 Map Conversion

In [13] it was investigated if it is possible to convert an occupancy grid map to an NDT map and they reported promising results. This approach was implemented in

this project and the pseudo-code can be seen in Section 2.5. Google Cartographer is used to get an initial occupancy grid map which is then converted according to Section 2.5 to an NDT map.

3.4.5 Refinement of NDT map

The map that has been converted from Google Cartographer can be improved. Since Google Cartographer builds the map without any prior knowledge about the environment the system can have rather large errors in the pose estimation. This leads to thicker walls since a larger error in pose leads to a larger error in the map and the precise details of the environment can be list. It is possible to use the map directly but to tighten the distributions in the cells where the data clearly describes a tighter distribution could lead to a better pose estimate. An approach to this is to simulate the same data that were used to build the map with Google Cartographer but with the localization algorithm that is used in this project. This map is continuously updated and after the simulation, the map is saved as a replacement for the initial map. The dynamic map that the localization algorithm builds also has occupancy probabilities which are used to determine which object should be used. This process is done with the new map a couple of times until the distributions have converged.

Algorithm 3 Replace static map with dynamic map

```
1: for all cell in NDT_Dynamic do
2:     if occupancyProbability(cell) < λ then
3:         Remove cell
4:     end if
5: end for
6: NDT_Static = NDT_Dynamic
```

In Algorithm 3 the distributions with low occupancy probability are removed and the static map is replaced with the cleaned dynamic map. λ is a constant of how sure the system has to be if a distribution is static and wants to use it. Note that the odometry has been used in this procedure, so the possibility of refining the map depends on odometry however the initial map from Google Cartographer with only the LiDAR sensor is possible to use for route planning. The refinement of the map can also be done using new data with odometry when an AGV has been deployed in the environment.

3.4.6 Automatically Clean the Map

The semi-static and dynamic object needs to be removed from the initial map since this will be the static map. One approach that is implemented is to manually remove the objects that is known to be semi-static or dynamic. This is easily done by marking which object that should be removed. However, manually cleaning the map might be time-consuming and it is easy to remove objects that seem to be semi-static in the map for example pallet racking. Because of this, an algorithm

was implemented that removes objects from the static map automatically based on previous saved dynamic maps. If an object is present in the static map and not in the dynamic map the object is removed from the static map. Algorithm 4 is based on the occupancy probability of the dynamic map. If the occupancy probability is under a certain threshold the object is removed. It is important to note that the cells in the dynamic are only considered if they have been denoted as occupied or not during the localization. Meaning that cells that have not had the possibility to be classified as empty or occupied, due to the cell being obscured, will not be taken into account when removing objects from the static map.

Algorithm 4 Remove objects with low occupancy probability

1: **for all** $\text{cell}_{Dynamic}$, cell_{static} in $\text{NDT}_{Dynamic}$, NDT_{Static} **do**
2: **if** $\text{cell}_{Dynamic}$ has been seen and occupancyProbability($\text{cell}_{Dynamic}$) $< \xi$ **then**
3: remove(cell_{Static})
4: **end if**
5: **end for**

In Algorithm 4 the pseudo-code for removing semi-static or dynamic objects is explained. The constant ξ is the threshold for removing the object.

3.5 Localization

In this section it will be described how the NDT map in Section 3.4 can be used to estimate the state of the AGV. While there are several filters capable of predicting the state of the AGV. The particle filter, also often called Monte Carlo Localization [23], is a common filter choice whenever a map is used. The particle filter can also handle non-linearities which makes it a suitable choice of filter. Since the objective is to track the 2D-position and the heading, (x, y, θ), also called pose, of the AGV, each particle will have an estimate of the pose.

In Figure 3.8, the overall structure of the localization can be seen. The blocks will in this Section be explained more thoroughly and how they are specifically utilized in the proposed implementation for this work.

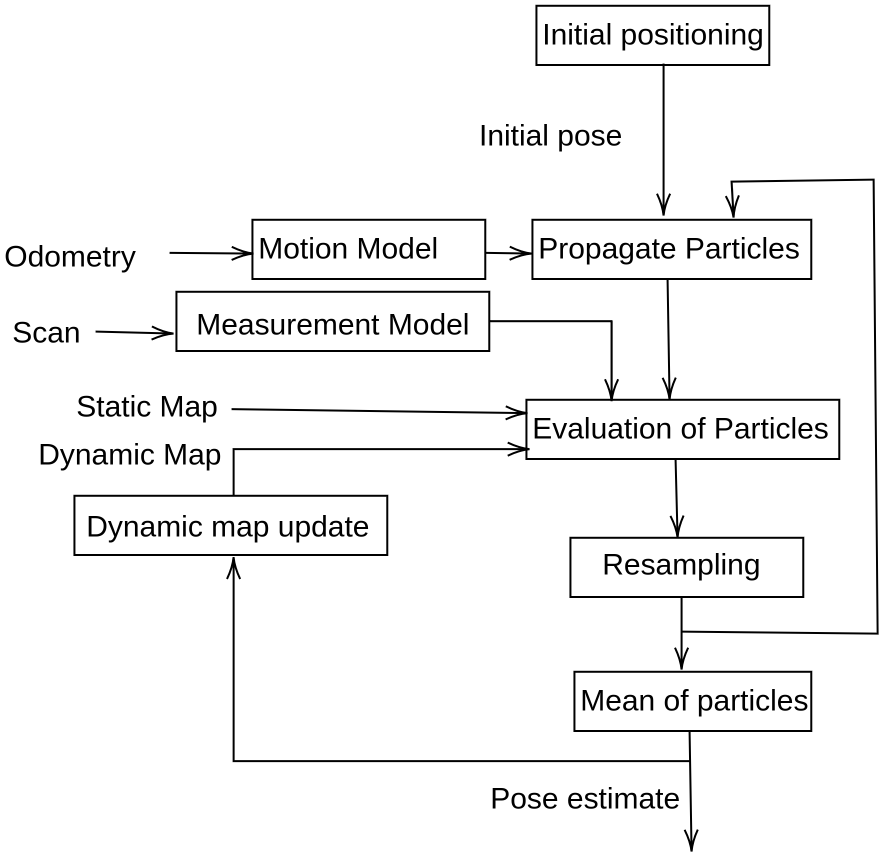


Figure 3.8: Block diagram over the localization.

3.5.1 Prediction

When localization is performed, it is important to use information from previous iterations of the localization in combination with measurements. In the case of this work, the AGV is equipped with wheel encoders providing odometry measurements to the system. This information given from the odometry can be used to propagate particles from one time instance to another and predict the new state. In the case of this work, the AGV is equipped with highly accurate wheel encoders which will be used to propagate the particles.

As mentioned in Section 3.1, the AGV used for this work has a differential drive motion model with encoders attached to each of the two driving wheels. How the motion model is applied to the position of the AVG can be found in (3.2) and (3.3). However, since the system is implemented using ROS where the part of the system that reads the information from the wheel encoders is set up as a publisher node, from here denoted as "odometry node", the odometry node itself does not know the current estimate of the state of the AGV. It is therefore instead chosen to update the state of the AGV in the odometry node starting from an arbitrary state and

using the odometry data to update this state.

Each time the state is updated, the state estimated by the odometry node is published on a ROS topic such that the node running the localization has access to the information when the particle propagation is made. What is then used in the propagation is instead the difference from the previous odometry state $\mathbf{x}_{t-1}^{odom}$ and the current estimated state $\mathbf{x}_t^{odom}$. Note here that the *odom* and later on WB superscript means that the pose are given in the *odom* or WB frame while no superscript indicates that the pose is given in the map frame.

$$\Delta\mathbf{x}^{odom} = \mathbf{x}_t^{odom} - \mathbf{x}_{t-1}^{odom} \tag{3.6}$$

Since the information used during the propagation of a particle is not actually the d_l and d_r values presented in (3.2) but instead, the difference between the previous state derived using odometry and the current one $\Delta\mathbf{x}odom$, seen in (3.6), the propagation must be updated accordingly. The previous state $\mathbf{x}_{i,t-1}$ of the particle i is known and is to be propagated using the data given from $\mathbf{x}_t^{odom}$ and $\mathbf{x}_{t-1}^{odom}$. Since the two states used from the odometry node $\mathbf{x}_t^{odom}$ and $\mathbf{x}_{t-1}^{odom}$ and the current state of the particle $\mathbf{x}_t^i$ are not connected to each other, even though they are both given at the same scale, the odometry state difference shown in (3.6) can not be directly applied to the particle state to propagate $\mathbf{x}_{i,t-1} \rightarrow \mathbf{x}_t^i$. What is instead done is to derive the difference in the state from $\mathbf{x}_t^{odom}$ to $\mathbf{x}_{t-1}^{odom}$ in the coordinate system of the AGV wheelbase, denoted as WB as seen in (3.7).

$$\Delta\mathbf{x}^{WB} = R \cdot \Delta\mathbf{x}^{odom}, \quad R = \begin{bmatrix} cos(\theta^{odom}) & sin(\theta^{odom}) & 0 \\ sin(\theta^{odom}) & -cos(\theta^{odom}) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{x}_t^{odom} = \begin{bmatrix} x^{odom} \\ y^{odom} \\ \theta^{odom} \end{bmatrix} \tag{3.7}$$

And the difference in state in the coordinate system of the wheel-base can then be applied to achieve the particle propagation $\mathbf{x}_{i,t-1} \rightarrow \mathbf{x}_t^i$ by rotating $\Delta\mathbf{x}^{WB}$ to the map coordinate system again using the particle direction as reference instead, as seen in (3.8).

$$\Delta\mathbf{x}_t^i = R \cdot \Delta\mathbf{x}^{WB}, \quad R = \begin{bmatrix} cos(\theta_i) & sin(\theta_i) & 0 \\ sin(\theta_i) & -cos(\theta_i) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{x}_t^i = \begin{bmatrix} x_i \\ y_i \\ \theta_i \end{bmatrix} \tag{3.8}$$

It is then possible to simply add the difference to the current particle of interest as seen in (3.9).

$$\mathbf{x}_t^i = \mathbf{x}_{i,t-1} + \Delta\mathbf{x}_t^i, \quad \Delta\mathbf{x}_t^i = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix} \tag{3.9}$$

Since the actual magnitude and characteristics of the error in motion model is difficult to estimate and can differ a lot depending on the environment where the AGV is located in, the error is chosen as a normally distributed error in x, y and θ with

a standard deviation based on the following equations:

$$\begin{aligned} \sigma_x &= e \cdot d + \epsilon_{x,y} \\ \sigma_y &= e \cdot d + \epsilon_{x,y} \\ d &= \sqrt{\Delta x_i^2 + \Delta y_i^2} \\ \sigma_\theta &= e \cdot \Delta\theta_i + \epsilon_\theta \end{aligned} \tag{3.10}$$

Where $\Delta x, \Delta y, \Delta\theta$ are defines as in (3.9) and e being the error ratio. The reason why $\epsilon_{x,y}$ and ϵ_θ are added is to ensure that the particles are able to perform a small random walk even when $\Delta\mathbf{x}_t^i = \mathbf{0}$ in order to be able to converge to an optimal state. The values used in the final implementation are $e = 0.1$, $\epsilon_{x,y} = 1$ mm and $\epsilon_\theta = 0.3$ degrees, based on observations.

3.5.2 Measurement Update

Once the particles have been propagated the estimated state of the particles are to be evaluated and the weights are then updated according to the evaluation. The system implemented in this work uses the DT-NDT map structure with a static and a dynamic map. While the system can also be used only using the static map, the hypothesis is that the addition of the dynamic map will increase the accuracy and robustness of the system, especially in highly dynamic or partially obscured environments. The observation used for evaluation is the measurements taken by the LiDAR sensor described in Section 2.1.3. The measurement model used for the system can be seen in Section 3.2 and takes the information given by the LiDAR sensor and provides the x and y coordinates of the measured points in the map coordinate system based on the current estimation of the particle position $\mathbf{x}_i$.

While the use of a dual time-scale map is based on findings made in [15], the calculation of the fitness value is instead based on (2.7) found in [10] and [12]. The reason behind this deviation from the suggested approach presented in [15] is based on poor initial results, potentially caused by lack of implementation details presented in [15]. This means that instead of building a local NDT map and then transform the map according to each particle p_i as seen in Section 2.7, the measurement model is used for each particle by itself and the value function is then applied to each measurement instead of each NDT cell in the local NDT map. The pseudocode for the particle evaluation algorithm implemented in this work can be seen in Algorithm 5.

Algorithm 5 Particle evaluation implemented

```
1: for all Particles i do
2:     Get observation z based on x_i
3:     F = 0
4:     for all measurement z_i in z do
5:         f_i = l(m_s, z_i)
6:         if f_i < ξ then
7:             f_i = l(m_t, z_i)p(c_i = 1)
8:         end if
9:     end for
10:    F = F + f_i
11: end for
```

With $l(m, z_i)$ being defined based on (2.7). In Algorithm 5, m_s denotes the static map of the DT map and m_t the dynamic map. As previously mentioned in Section 2.7, $p(c_i = 1)$ denotes the occupancy probability of the cell c_i.

The value function finds the corresponding cell where the measurement appears to be. However, in Section 2.4.2 it was mentioned that checking the neighboring cells could give a better estimate. This is used by instead of only checking the corresponding cell to the measurement the 3×3 cell around the specific cell are checked for a better fitting distribution.

Comparing to Algorithm 2, the suggested Algorithm 5 would replace lines 3-13 in Algorithm 2 while the rest of the steps in the pseudocode remains the same at a pseudocode level. Now that the fitness value F for each particle has been derived the weights of the particles can be updated. The value function gives a fitness value F which is used to update the weight according to (3.11)

$$w_t^i = w_{t-1}^i \cdot F \tag{3.11}$$

where w_t^i is the weight at time t for particle i with state $\mathbf{x}_t^i$. The weights of the particles can then be used to perform resampling. A common initialization of the weights is to set:

$$w_0^k = \frac{1}{N} \tag{3.12}$$

Where N is the total number of particles used in the particle filter.

3.5.3 Resampling

Once all of the weights have been updated, the resampling takes place. As mentioned in Section 2.6.1, the reason for resampling is to focus the collection of particles in areas of the previous estimations that are evaluated to higher fitness values, given by Algorithm 5. This process ensures that particles with higher fitness values are duplicated while particles with lower fitness values are removed, which is done without increasing the total number of particles. In the proposed implementation residual resampling was used [24]. It is not necessary to resample the particles each time the

weights are updated and a criterion on when to resample could potentially be based on the spread of the particles. One reason for not resampling after each update is to reduce the computational complexity however since computational complexity was not a priority of this work, the resampling was implemented to take place after each update of the weights. Also note that for the implementation presented in this work, the time to resample the particles was negligible in comparison to the time needed for other parts of the system such as evaluating particles.

3.5.4 Time Delay

As described throughout this section and visualized in Figure 3.8, a LiDAR observation is given and the odometry up until the time of the observation is used to propagate the particles. All of the particles are then evaluated, resampled and the mean of the particles are taken to get a single estimate of the state of the AGV. Meaning that the estimated pose is the pose of the AGV at the time of the measurement. However, since there is a non-neglectable time complexity for the system there is now a time delay between the current real-time and the time that the estimate is based on. Since it is important for the control loop to have the current estimated pose the time delay has to be removed. A real-time control loop that tries to adjust the pose based on an older estimate could lead to an unstable system which could lead to fatal errors.

The way this was solved was to introduce a final step before giving the estimated position to the AGV controller. The final step consists of propagating the estimate using the odometry recorded up until the current real-time. This way the estimate that is given to the controller is based on the previous estimate from the particle filter and propagated with the odometry without any additional error compared to Section 3.5.1.

An important note is that the odometry data used for propagating the estimated position is later on used again at the original propagation step since the particles themselves are not propagated in the new final step, only the combined estimated state of the AGV is propagated.

3.5.5 Dynamic Map Update

Since the environment changes and objects are moved around continuously the dynamic map has to be updated with the latest observations, both static and semi-static. However, the static objects in the dynamic part will not be used since the static map has priority over the dynamic part. The static map will be constant during navigation while the dynamic map will change online.

Each cell of the dynamic map has a distribution and an occupancy probability as mentioned in Section 2.7. Both the distributions and the occupancy probability have to be updated with the latest surroundings. In Section 2.7.1 it was explained that it is not possible to store all the measurement points due to memory complexity and computational complexity when recalculating the distributions in the cells.

Therefore the distributions is updated without storing all the old data using (2.18) and (2.19). Again note that while (2.15) and (2.16) are able to precisely derive the updated mean and covariance for a given cell without the need of storing all of the data used to calculate the current mean and covariance. The approach presented ins (2.18) and (2.19) allows cells to adapt to the changing environment with a variable adaptation rate based on the value of M as presented in (2.20).

An important note is that the implementation suggested in this work does not use the particle evaluation as in [15], described in Section 3.5.2. However, the creation of a local NDT map based on the current observation and estimated state is still used when updating the dynamic map. While (2.18) and (2.19) could be applied using singular measurements one by one, there is a problem doing it that way. The mean and covariance of a cell are not the true mean and covariance given all data but instead a regency weighted mean and covariance estimate. Therefore, the sequence in which data is added to the mean and covariance estimate is relevant to the resulting estimate. Taking a wall segment present in a cell as an example, the surface of the wall can only be seen from a 180 degrees viewing points. And since the data from the LiDAR sensor is given in a rotational sequential order, the order in which the wall inside of the cell is measured will always be the same. If the mean and covariance of the cell were then updated using singular measurements, the last measurement added that measures the wall in the cell would have a larger weight compared to the first measurement even though they were taken in the same observation. While the effect of this weight inconsistency might be very small given that $M >> 1$ in (2.20), the choice of adding an observed NDT map to the current dynamic NDT map instead of single measurements was based on this. Another difference made in comparison to [15] was to slightly modify the proposed (2.17) when updating the log-odds for a cell. When a cell is being denoted as empty for a long period of time, the log-odds will grow towards negative infinity meaning that the cell must then be denoted as occupied for a very long time before the occupancy probability will take a meaning-full non-zero value. To make the dynamic map react a bit faster to changes, a lower limit of the log-odds value for each cell was added according to (3.13).

$$p_{log}(c_i = 1) = \begin{cases} p_{log}(c_i = 1), & if\ p_{log}(c_i = 1) > p_{log,min} \\ p_{log,min}, & otherwise \end{cases} \tag{3.13}$$

Where $p_{log,min}$ is a tuning parameter based on the dynamic nature of the environment, in this case set to -6 for each cell. Since the same is true for an object being present for a long time and then removed, a similar limit was implemented as an upper limit with the parameter $p_{log,max} = 6$

3.5.6 Initial Positioning

In order to have a prior knowledge of the state of the AGV, an initial pose needs to be determined. This has to be done using only the LiDAR sensor since it is not feasible to drive the AGV automatically without knowledge of the state of the AGV. The initial positioning then needs to be done given a LiDAR observation and

the map. In this work, only the static map is used since the environment could potentially have changed since the last time the AGV was used.

Algorithm 6 Initial positioning

```
Initialize particles in the environment
for 10 iterations do
    for all Particles p_i do
        Propagate particles using random walk
        Evaluate particles modified only static map
    end for
    Normalize weights
    Resample
end for
Initialize particles around best particle
for 10 iterations do
    for all Particles p_i do
        Propagate particles using random walk
        Evaluate particles only static map
    end for
    Normalize weights
    Resample
end for
```

In Algorithm 6 the pseudocode for determining the initial state can be seen. The initial positioning is based on the localization algorithm previously described in Section 3.5. A particle filter localization approach is used but no odometry is used. The algorithm is divided into two parts, the first part finds the global correct state and the second part refines this state.

The number of particles is increased significantly compared to when the prior state is estimated since now the particles are distributed in the entire environment. The particles are sampled uniformly in the environment separately. When a particle has been sampled the measurement model gives the positions of the hits from the laser rays given the LiDAR scan and the position of the particle. The measured points based on the position of the particle are then derived, if less than 95% of the points are inside the environment the particle is resampled randomly until more than 95% of the points are inside the environment. This is done for each particle. This ensures that all initial particles have a reasonable chance of being an accurate estimate of the pose of the AGV.

Once all the particles have a feasible state they are evaluated using a modified value function that only counts how many cells with distribution the scan hits. This is done because the heading of the AGV is difficult to sample correctly and a small change in heading has a larger effect on the fitness value based on the probability of the cells. This problem was seen to be mitigated using only a binary hit or miss value for the initial positioning evaluation.

When all the particles have a fitness based on the number of hits, the respective weights are updated and normalized. The particles are then resampled just as in Section 2.6.1 and propagated using a random walk. These steps are then repeated a number of times and instead of taking the mean of the particles with the highest weight is used in the next part. The reason for choosing the best particle instead of the mean of all particles is because the particles often group in multiple possible states throughout the environment and instead of iterating until all particles have converged to the same location the best particle is used.

The next part of the initial positioning is to refine the state of the best particle. A number of new particles are sampled around the best particles. These particles are propagated using a random walk. The particles are then evaluated with the value function used in localization described in Section 3.5.2. The best particle is used as the prior state to the localization algorithm.

4

Experiments

The future vision is to be able to replace the current reflector-based system, explained in Section 2.1.1, with the implemented system in this work. It is therefore important to be able to compare the two. However, comparing the two systems is a non-trivial task when there is no ground truth to compare to. The current reflector-based system is not accurate enough to be used as ground truth, a visual example of this can be seen in Figure 5.8 and Figure 5.9. The system will be tested in scenarios that are similar to the ones expected in a long term use of the system

4.1 Parameter Study - Optimal System

In order to arrive at the optimal implemented system, the system itself must first be compared using different setups. There are multiple parameters in the system, however, most of them are decided based on observations throughout the implementation. The parameters chosen to further evaluate are the following:

- Will the search space have a positive effect on the localization? The search space was first introduced and explained in Section 2.7.
- Should the dynamic map be used during localization or should only the static map be used?
- Two important parameters:
  - What is the optimal number of particles?
  - Is there any reason to not use all measurements?

The offline experiment will be to drive the AGV in an ordinary situation with no extra additional difficulties. The route for the experiment can be seen Figure 4.1.

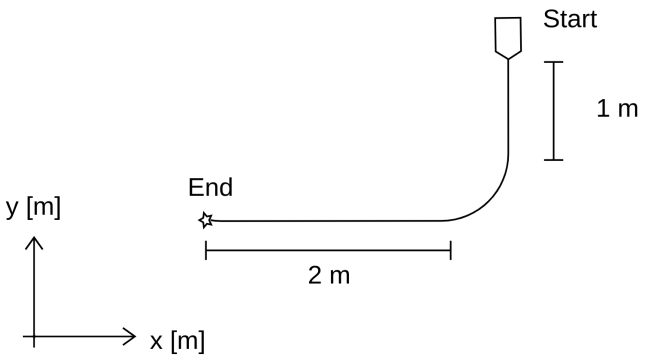


Figure 4.1: Route automatically driven while recording data. The AGV drives from the start position, in a forward direction to the end position and the data recording is then stopped.

Since there is no accurate ground truth available, the system is compared against itself using the same data recorded using ROS for each simulation that is compared towards each other, with the current reflector-based system being used for localization when recording the data. This is done by first running an offline sequence where the system is assumed to be in its optimal setup. The search space is assumed to never have a negative effect on the system and is therefore used, all measurements from each observation are used as well as a large number of particles. By then removing the use of the search space and slowly decreasing the number of measurements and particles, the system can be evaluated towards itself using the root mean square error (RMS error) of the pose estimate.

4.2 Parameter Study - Real-time System

Since time complexity was not a priority throughout the implementation, the implemented system is unable to use all data with a sufficient number of particles with or without using search space at the same time. The number of particles used and the number of measurements used for each observation must be evaluated when taking the possible update frequency of the system into account. By using a subset of the parameters used in the offline experiments, a new similar experiment will be conducted where also the update frequency is changed between offline localization sequences on the same data presented in Section 4.1. By then testing and noting down the actually possible frequency using different combinations of the parameters, the optimal choice is assumed to be the one giving the lowest error when compared to the optimal setup similar to the offline experiments presented in Section 4.1.

4.3 Online Experiments

In the online experiments, the repeatability of the system will be tested by measuring the stopping position of the AGV when navigating the same route as presented in Figure 4.1 multiple times by letting the AGV navigate back to the start again and then navigate the original route again. The current reflector-based system and new proposed systems will individually be providing the AGV with localization. External laser meter sensors will be used to measure the position of the AGV as presented in Figure 4.2

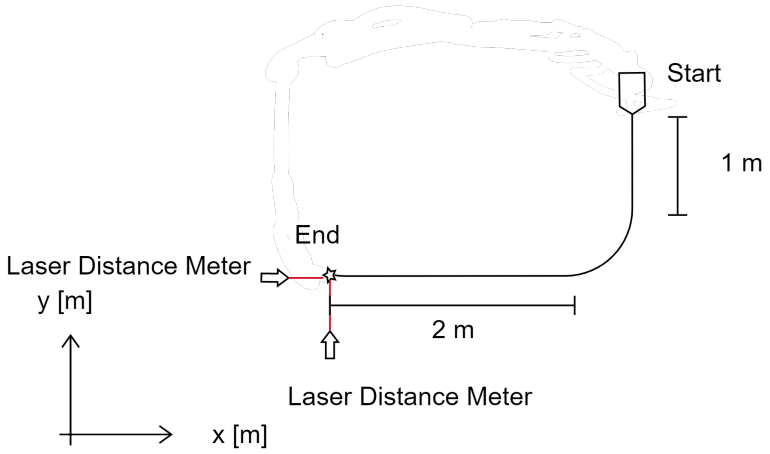


Figure 4.2: Sensor setup, the AGV is driven similarly to Figure 4.1 however the AGV is also automatically driven back to the start position and starts over again.

This way, the current reflector system, the new system with only the static map, and the new system with the DT map can be compared. The external measurements cannot be evaluated with a ground truth but the variance in position can be compared. Also, the difference in position between the system using the static map and the system using the DT map can be compared since they are using the same coordinate system. The external sensors used are the Leica DISTO D2 with an absolute measurement error of ± 1.5 mm with a repeatability error assumed to be much lower based on testing.

5

Results

In this Section the results from the Experiments in Section 4 will be presented. The results have four separated parts which are, static map creation, initial positioning, offline localization, and online localization. The results regarding the map creation are difficult to evaluate since there is no ground truth map available, the localization results are however heavily dependent on the quality of the map and the result can therefore be implicitly evaluated based on this.

5.1 Static Map Creation

This section shows the results from the static map creation. Data from the real sensors are recorded and used to create the static map. The occupancy grid from Google Cartographer can be seen in Figure 5.1. The occupancy grid is built using the method in Section 3.4.2 with a grid size of 5×5 cm.

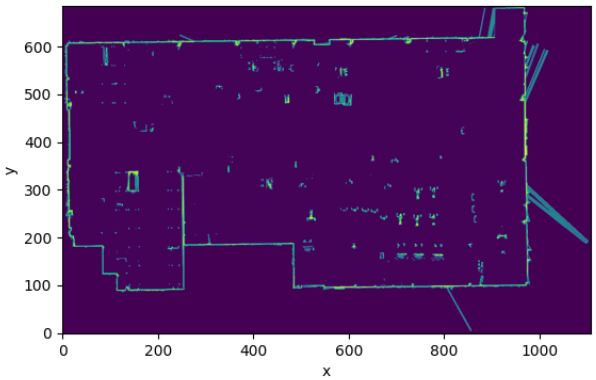


Figure 5.1: Occupancy grid from Google Cartographer, where one unit is 0.05 m

The initial occupancy grid is rotated to the coordinate system that the company usually uses and this can be seen in Figure 5.2. The reason why the coordinate system is changed to the one Soft Design uses is so that navigating predefined routes is possible using both systems.

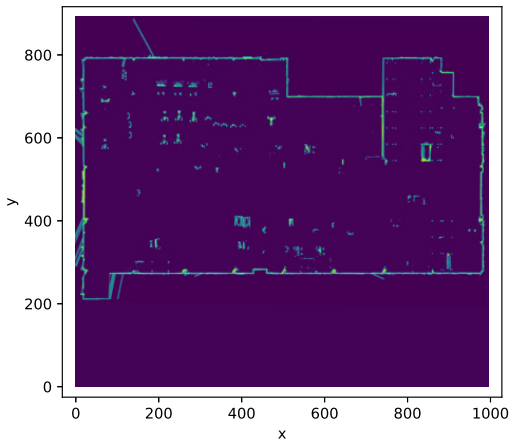


Figure 5.2: Rotated occupancy grid to wanted coordinate system

The rotated occupancy grid is converted to the NDT structure using the method in Section 2.5.

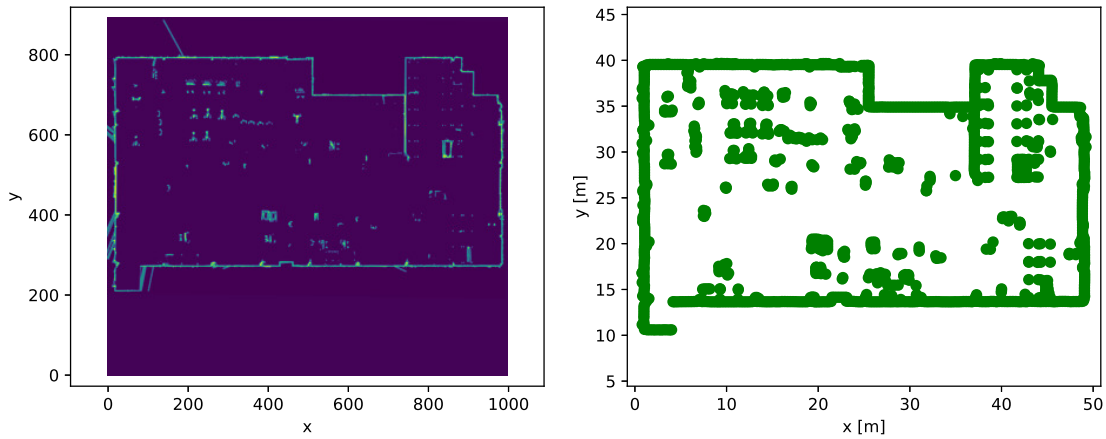
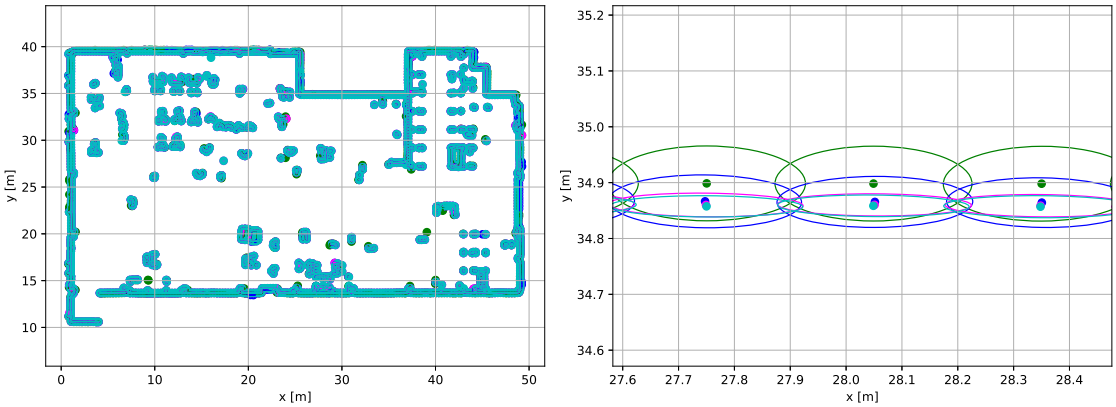


Figure 5.3: NDT map from Occupancy grid, with a grid size of 0.05 m. The occupancy grid generated by Google Cartographer is seen on the left and the converted NDT map on the right.

As can be seen in Figure 5.3 there are walls and there are objects in the middle. These objects are parked AGV's and other semi-static objects. These semi-static objects needs to be removed to make the static map only contain static objects. The conversion from an occupancy grid to NDT works, however, the distributions could potentially be a bit more accurate where there are walls. This is because the distributions in the axis along the wall are preferable to be more confident. This refinement of the map is performed as described in Section 3.4.5.

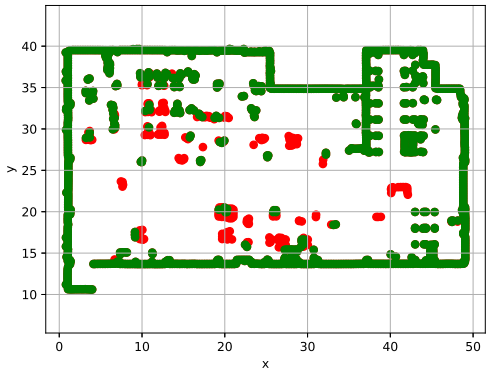


(a) Refine the distributions **(b)** Refine the distributions zoomed

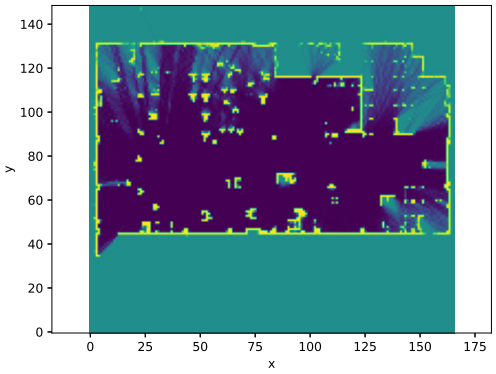
Figure 5.4: Three simulations to updated the map, green is the map converted from Cartographer, blue the first refinement, pink the second refinement and cyan the third refinement.

As can be seen in Figure 5.4 the distributions become tighter and moved closer to the sensor, this is due to the fact that Google Cartographer builds a map with thicker walls. The reason that Google Cartographer builds thicker walls is that in the initial usage the pose estimation has some error and builds walls that are thicker, and once the pose estimation has improved, Google Cartographer cannot remove the occupancy probability since the cells behind the true wall can not be seen. The reason it is possible to use the dynamic map as a replacement for the static is because the localization becomes more accurate when localizing using a more accurate map. It is important that the map represents how the sensor sees the environment.

To remove the semi-static object, the method in Section 3.4.6 was performed. In Figure 5.5 semi-static object from one time instance is removed such that the objects that has been moved since the initial map creation is removed from the static map and the object that is still present from the initial map creation is not removed. As can be seen in Figure 5.5a the red distributions is removed based on the dynamic map in Figure 5.5b. This procedure is performed several times with dynamic maps from different time instances where the environment has changed in between. When all semi-static objects have been moved since the initial recording of the environment the static map can be updated such that it only contains static objects.



(a) The green and red is the static map and the red is the objects that are suggested to be removed.



(b) The occupancy probability of the dynamic map of a DT map that was used in this iteration to remove the objects.

Figure 5.5: Automatically remove semi-static objects from the static map using a dynamic map from another time instance.

Using the dynamic maps from 5 time instances over 3 weeks the static map made it possible to automatically clean the static map as can be seen in Figure 5.6.

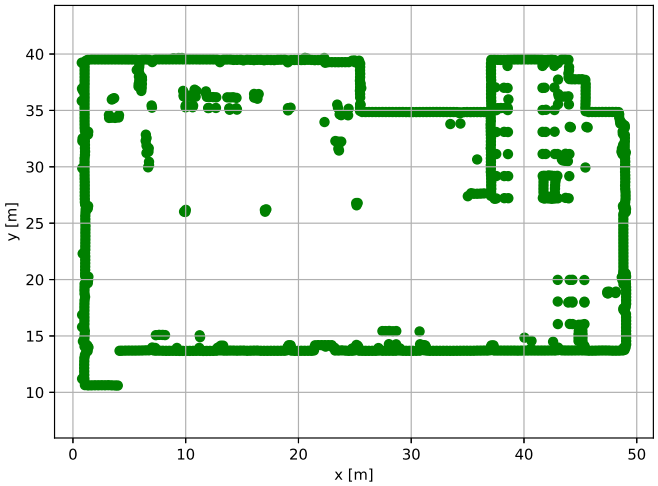


Figure 5.6: Automatically cleaned static map with 5 dynamic maps over 3 weeks.

Note that the goal of automatically cleaning the map is to be able to never use any manual cleaning. This would most likely require the system to operate using a non-optimal static map for the initial usage of the system. While this has not been tested in any repeatability test, the dynamic map leading to the map seen in Figure 5.6 where all created based on the initially created map seen in Figure 5.4 indicating that the system is at least capable of localizing using the initial map without failure. As mentioned, the repeatable accuracy is, however, not measured during this process.

After the map has been both cleaned manually and automatically the final static map is ready to be used. In Figure 5.7 the final initial map can be seen where all the semi-static and dynamic objects are removed and there are only static objects left.

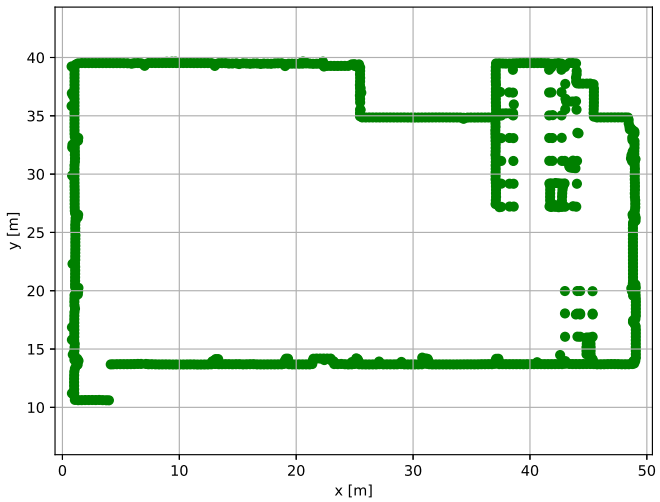


Figure 5.7: The final static map after both automatically and manually cleaned.

5.2 Initial Positioning

By running the initial positioning algorithm on data where the true pose of the AGV is known, the success rate of the initial positioning can be derived. The known pose of the AGV is found by running the sequence of data through the system offline and saving the estimated pose at each observation. This means that the estimated pose has a certain uncertainty but will be globally correct. The estimated pose from initial positioning can then be compared to the corresponding pose for the observation and using a threshold in distance from the estimated true pose a success rate can be found.

The time to estimate the initial pose is heavily based on the number of particles in the initial step of the algorithm described in Section 3.5.6. And the number of particles needed is heavily dependent on the size of the environment. In Table 5.1 the success rate and time consumption for a single initial positioning based on the number of particles used for the initial initialization of particles described in line 1 of Algorithm 6 can be seen. The success rate and average time consumption are based on tests where the AGV is placed at 10 separate locations in the environment and 5 initial positioning attempts are performed for each location on separate LiDAR observations, making up a total of 50 attempts.

Number of particles used	Average time consumption	Success rate
500	6s	0.32
2000	22s	0.52
5000	53s	0.70
10000	110s	0.78
30000	320s	0.80

Table 5.1: Success rate and time consumption of the initial positioning based on number of particles used for first initialization of particles.

5.3 Offline Localization Results

The route used to derive the offline localization experiments can be seen in Figure 4.1 and the trajectory, estimated by the current system as well as the two optimal systems using static and dual-time maps can be seen in Figure 5.8 where DT stands for Dual-Timescale.

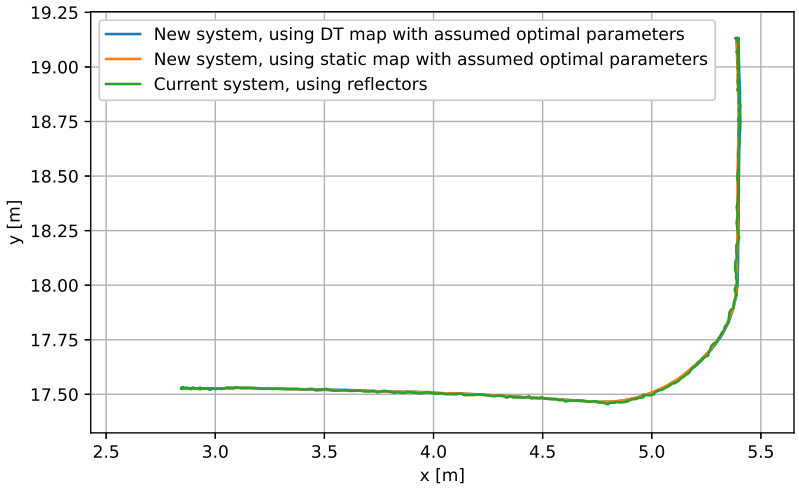


Figure 5.8: Route automatically navigated by the AGV used for offline simulations. The assumed optimal parameters are based on the optimal setup presented in Section 4.1

A zoomed-in version of Figure 5.8 can be seen in Figure 5.9 which clearly demonstrates that the current system can not be used as a ground truth due to its highly stochastic nature. Another important note is that there is a small difference in the overall trajectory between the current reflector-based system and the proposed system. This difference can not be solved by translation or rotation the two coordinate frames, and according to the company the error might very well lie in the reflector-based system since a small constant bias in pose estimation is not a big problem as long as it is constant. The reason for the difference between the dual-time system and the static system is unknown however the difference is rather consistent

over multiple runs even when varying the number of particles and number of measurements used for each observation. Because of this, the static system will in the offline simulations only be compared to the static so-called optimal setup and the dual-timescale system will only be compared to the dual-timescale optimal setup.

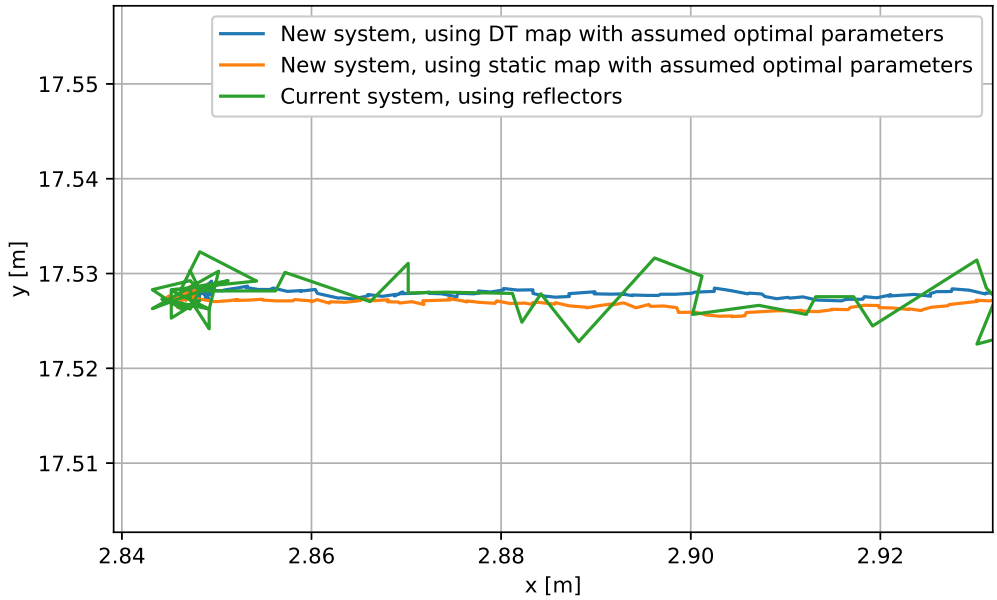
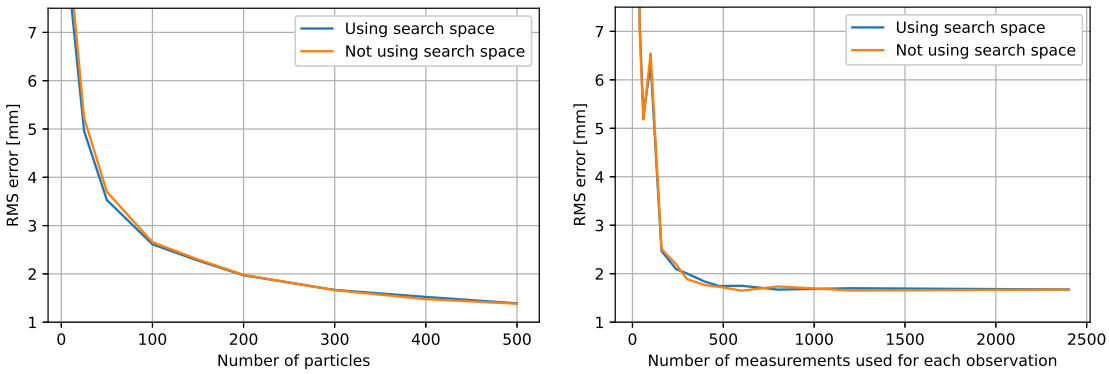


Figure 5.9: Zoomed in version of Figure 5.8 showcasing that the current system is infeasible to use as ground truth. As mentioned the reflector-based system has a updated frequency of 8 Hz and the new proposed system has a maximum update frequency of 35 Hz based on sensor limitation

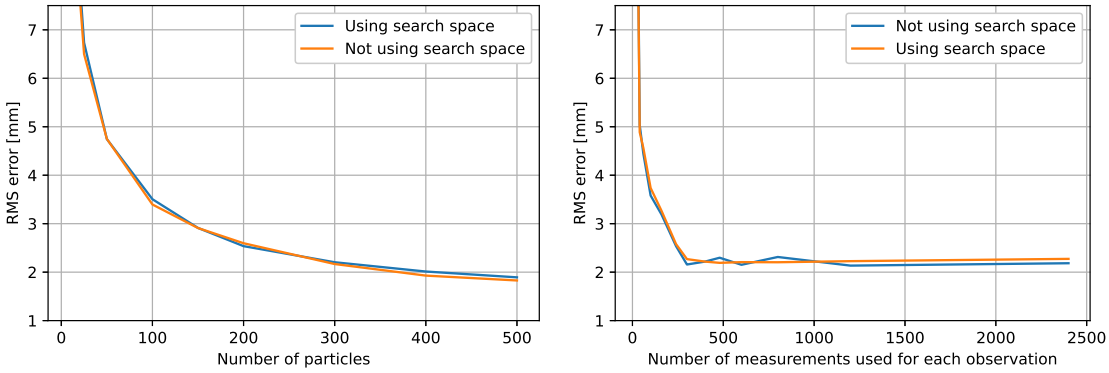
The offline experiments were conducted as presented in Section 4.1. Namely to vary the use of search space or not, the number of particles, and the number of measurements used for each observation for the static system and the dual-timescale system individually. For visual purposes, the particles used and the number of measurements per observation are split up into two figures for the static and dual-timescale systems individually. The result will be presented here and then analyzed in the sections to come.



(a) Varying the number of particles.

(b) Varying the number of measurements used for each observation.

Figure 5.10: Offline localization results from using only the static system.



(a) Varying the number of particles.

(b) Varying the number of measurements used for each observation.

Figure 5.11: Offline localization results from using the dual-timescale system.

As mentioned in Section 4.1, the error, given as root mean squared error, is derived by comparing the estimated trajectory to the assumed optimal trajectory presented in Section 4.1.

5.3.1 Using Search Space

As can be seen in all of the subfigures in Figure 5.10 and Figure 5.11, the use of search space made little to no difference in the error compared to the optimal system where search space was used. Since the search space, as presented in Section 2.7, looks at a 3×3 grid of cells (9 cells in total), instead of just one cell, the time complexity is much worse when using the search space. It is therefore from here on chosen to not use the search space.

5.3.2 Varying Number of Particles

Taking a look at Subfigure (a) in both Figure 5.10 and Figure 5.11. It can clearly be seen that the number of particles used is highly relevant for the accuracy of the localization. Since the error seems to be constantly decreasing as the number of particles increase, no further conclusions can be made other than that a larger number of particles is always preferred when time complexity is not taken into account. It is important to note that the optimal system, serving as ground truth for the evaluation, uses 1000 particles. It is however not feasible to use more than 500 particles for the real-time system due to hardware limitations.

5.3.3 Varying Number of Measurements Used for Each Observation

While the error seemed to always decrease as the number of particles increased within the chosen interval, the same can not be said for the number of measurements used for each observation. In Subfigure (b) in both Figure 5.10 and Figure 5.11 it can be seen that increasing the number of measurements used for each observation seems to decrease the error up to around 500 measurements for each observation for both the static and the dual-timescale system, but not after. In future evaluation and results, the number of measurements used for each observation will therefore be limited to a number slightly larger than 500, chosen as 600 measurements used for each observation. In other words, for the coming tests, the number of measurements used per observation will be in the interval of $(0, 600]$.

5.4 Choice of parameters for real-time system

I Section 5.3 it was concluded that the search space should not be used. The number of particles should not be above 500, based on hardware limitations. And the number of measurements used for each observation should not be above 600 since it from there gave no improvement in accuracy while increasing the time complexity for the system. Since both the number of particles and the number of measurement used for each observation has a large effect on the possible update frequency for the real-time system, they must both be tested in combinations in order to arrive at an optimal choice. In order to derive an error based on the update frequency of the system, similar offline localization sequences were performed as in Section 5.3 with the addition of only using a subset of the LiDAR observations, effectively simulation the decreased frequency of the real-time system. In Figure 5.12 and Figure 5.13, the error based on the frequency for different combinations of parameters for the static and dual time-scale system can be seen.

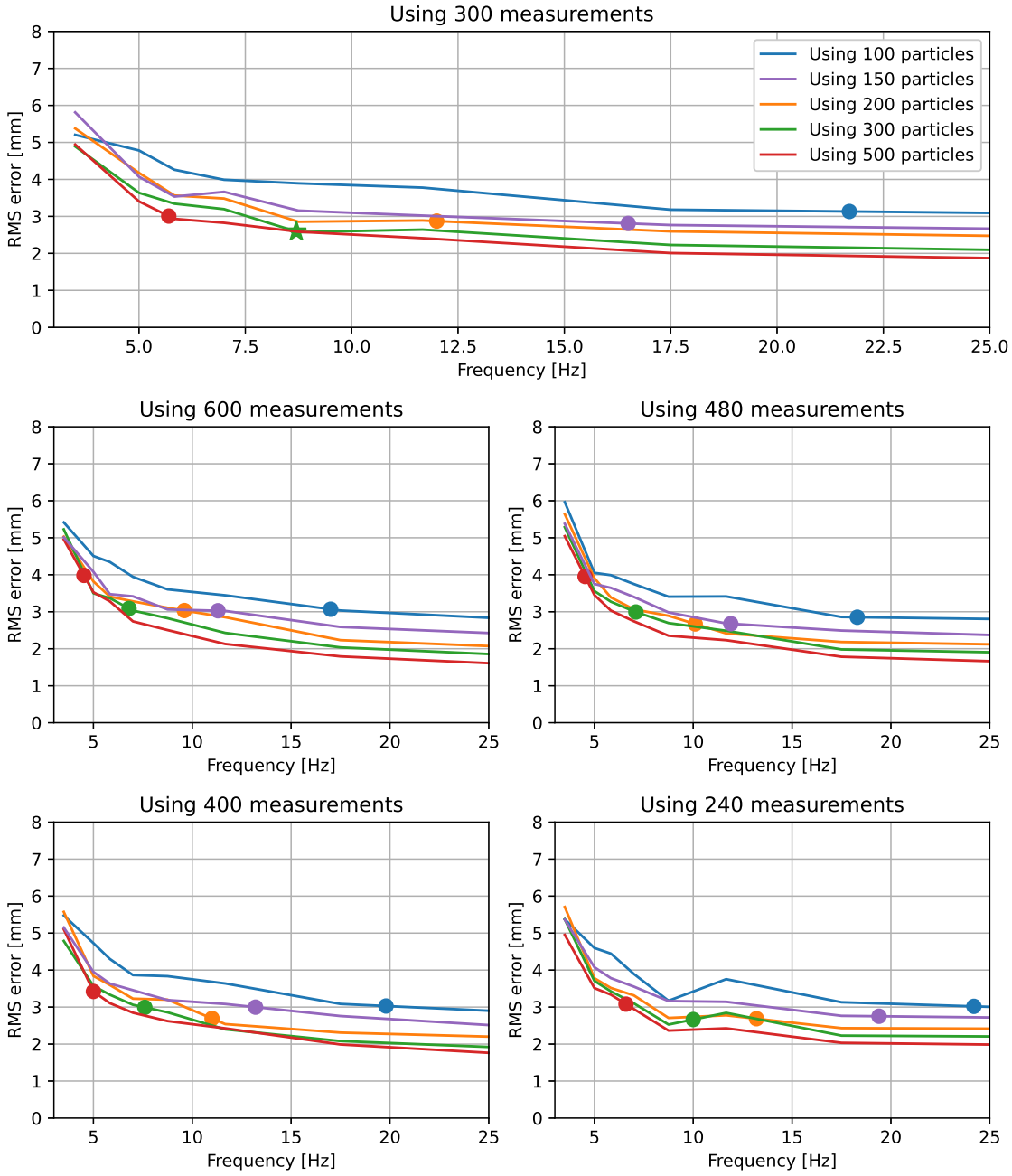


Figure 5.12: Error based on frequency for the static system. The possible real-time frequency is marked as a dot and the lowest possible error given the possible frequency marked as a star. Different subfigures uses different values for measurements per observation.

From Figure 5.12 it can be concluded that the optimal choice of parameters is, for the static system, to use 300 measurements per observation and 300 particles. Based on the evaluation method presented in Section 4.1. Which should give a frequency of approximately 8.7 Hz when used in real-time.

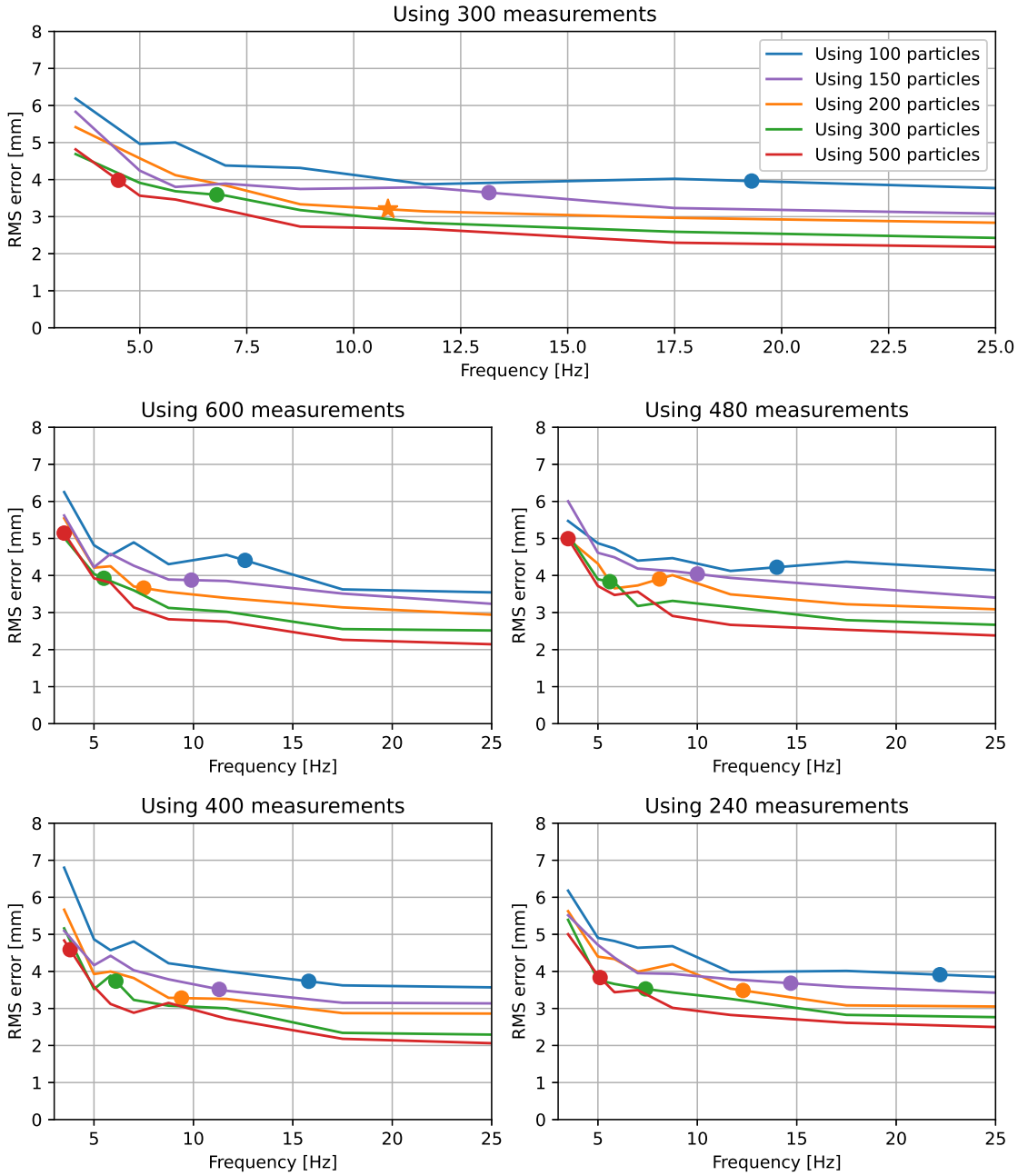


Figure 5.13: Error based on frequency for the dual-timescale system. With results structure the same way as in Figure 5.12

Analyzing Figure 5.13 in a similar fashion as for Figure 5.12, it can be concluded that based on the hardware limitations, for the dual-time system, using 300 measurements per observation and 200 particles gives the lowest RMS error based on the evaluation method presented in Section 4.1. Giving a frequency of approximately 10.7 Hz. Note that the maximum frequency, based on the limitations of the LiDAR is 35 Hz. It was however consistently found that a lower frequency, enabling more measurement per observation and particles to be used gave a lower error.

A comparison between the proposed real-time parameters with the achievable frequency and the current reflector-based system can be seen in Figure 5.14, similar to the visual comparison in Figure 5.9.

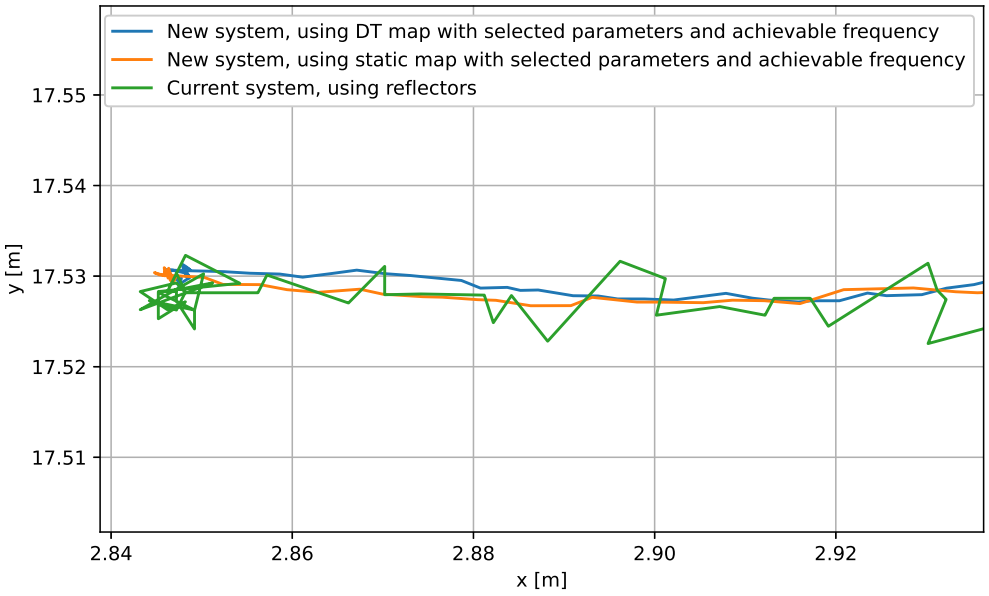


Figure 5.14: Zoomed in version of the trajectory when the current reflector based system and the proposed real time system with achievable frequency, reflector-based system: 8 Hz, static system: 8.7 Hz, dual-timescale system: 10.7 Hz.

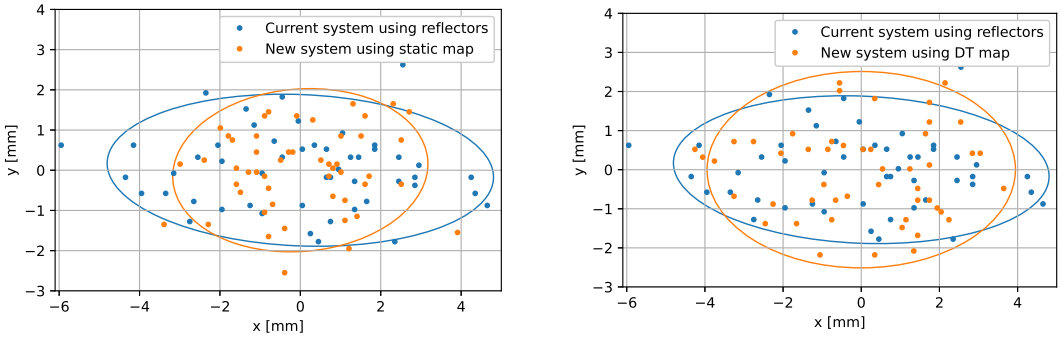
5.5 Online Localization Results

In the online tests, the AGV was automatically driven the route in Figure 4.2 which is the same as the offline tests. The position of the AGV was be measured with two external laser distance meter to be able to measure the actual position of the AGV, as presented in Section 4.3. Repeatability is the most important result for the localization, namely that the AGV is able to pick up a pallet or similar at the same position every time.

The test is divided into two parts, in the first part the environment is not intentionally changed during localization, and in the second part is the environment changed such that an important wall is covered.

5.5.1 Repeatability without environment change

In this testing the environment is not drastically changed, however, the environment is the production and warehouse at Soft Design, and people and other AGV:s are constantly moving around, making the environment highly dynamic.



(a) System using the static map **(b)** System using the DT map

Figure 5.15: Repeatability test without environment change. The dots are the measurements and the corresponding ellipses are the 2σ-level deviation.

In Figure 5.15 there are 50 different measurements for each system. The current system that is using reflectors is moved to zero mean because the current system is using a different coordinate system such that the conversion is arbitrary. The measurements from the static system and the dual-time system is also moved to zero mean individually.

	Static system	Dual-Time system	Reflector based system
Longitudinal mean	0	0	0
Lateral mean	0	0	0
Longitudinal std	1.6	1.9	2.4
Lateral std	1.0	1.3	1.0

Table 5.2: Results when not changing the environment in [mm]

As can be seen in Figure 5.15 and Table 5.2 the static system has a lower or equal standard deviation in both lateral och longitudinal axis compared to the reflector-based system. The standard deviation of the system using dual-time is slightly higher than the static. While the reflector-based system has the equally best standard deviation in the lateral axis, it has the highest and worst in the longitudinal axis.

Overall the system using static is giving the best results, however, all three systems are comparable and have sufficiently high accuracy to be used in production. It is however important to note that a slight difference in the actual mean value of the new system using the static and DT map was noticed, the original measurements from the sensors can be seen in Figure 5.16.

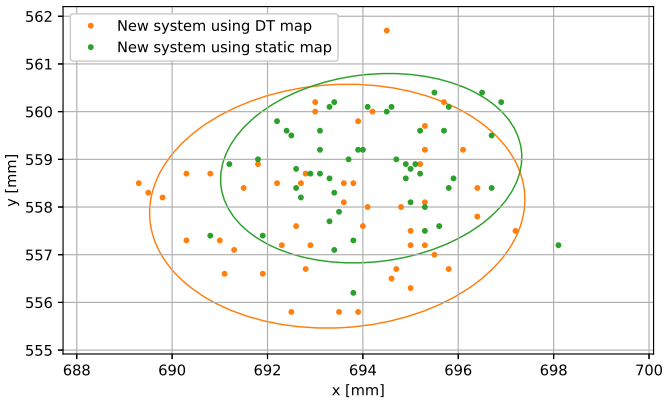


Figure 5.16: Actual measurements, not moving to zero mean for the new system using static and DT map respectively.

The reason for this deviation is unknown, however, it is consistent with the result presented in Figure 5.9. It is also important to note that the deviation is very small and that the sample size of 50 measurements might be too small to make any further conclusions.

5.5.2 Repeatability test with environment change

This test is a continuation of the previous test. A wall near the route is covered such that the LiDAR cannot see the static wall, this can be seen in Figure 5.17. The AGV drives the route 50 times with each system setup. In Figure 5.18 the 50 measurements for each system are presented. The measurements of the static system and the dual-time system are moved with the same mean as in Figure 5.15. Note that the new wall introduced is placed far enough away from the existing static wall such that it only blocks the vision of it, it does not occupy the same NDT cells.

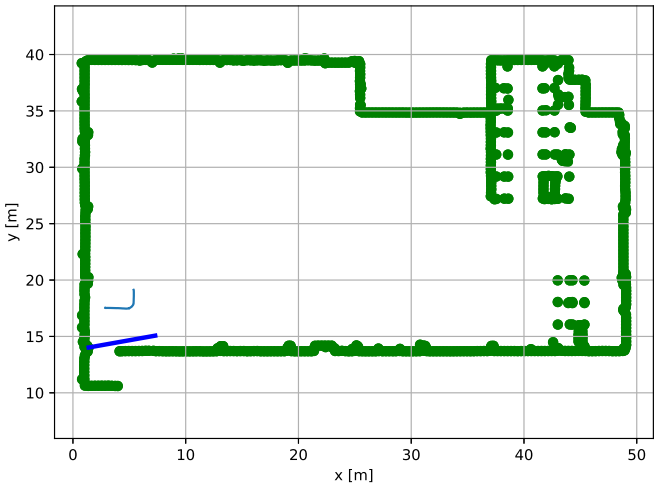
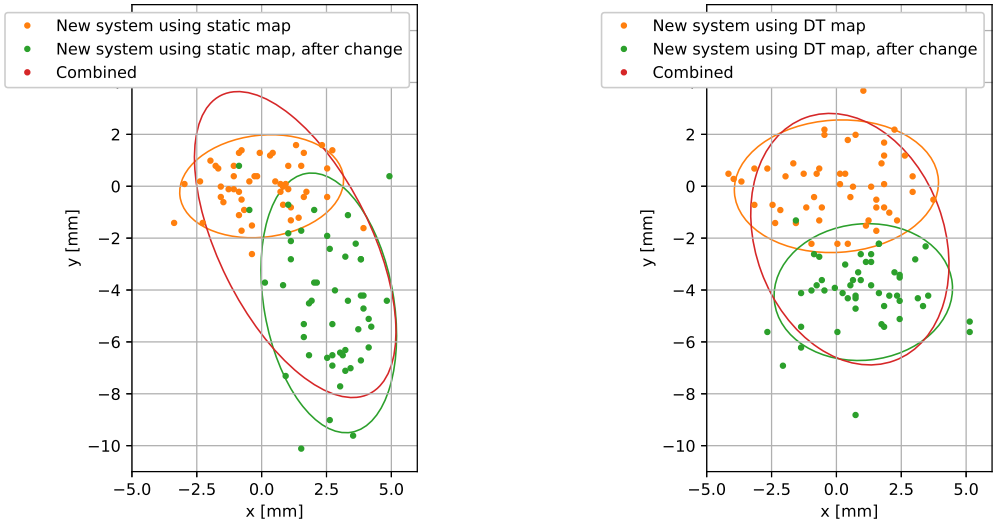


Figure 5.17: The environment where the green is the static map, the blue is the added wall, light blue the route



(a) System using the static map

(b) System using the DT map

Figure 5.18: Repeatability test when the environment has been changed. The dots are the measurements and the corresponding ellipses are the 2σ-level deviation.

	Static map		DT map	
	After change	Combined	After change	Combined
Longitudinal mean	2.6	1.3	1.0	0.5
Lateral mean	-4.5	-2.2	-4.1	-2.0
Longitudinal std	1.3	1.9	1.7	1.9
Lateral std	2.5	2.9	1.3	2.4

Table 5.3: Results when changing the environment in [mm]

In Figure 5.18 and Table 5.3 it can be seen that the system using dual-time and the system using static has an offset from the previous test seen in Figure 5.15. The system using static has the highest standard deviation in the lateral axis. This is because the wall the system cannot see is in that direction. The system using the DT map has a similar standard deviation in the lateral axis and slightly lower in the longitudinal axis. The combined measurements are the data from both tests where it can be seen that the system using the DT map has a lower variance than the system using only the static map. The current reflector-based system failed to localize in the environment because of the fact that one reflector was blocked by the new introduced wall.

6

Conclusion and discussion

In conclusion, the work has managed to develop a system that, from no prior knowledge of the environment, builds a map of the environment and uses it to localize with high enough accuracy to be used in production. The initial positioning is, however, not sufficiently robust and needs to be changed to be used in production. The initial positioning in this work is developed with the approach that the system has no prior knowledge about the pose of the AGV. If the system would have some knowledge about where in the environment the initial algorithm would have a higher success rate.

6.1 Static map creation

In Section 5.1 the results from the static map creation were presented. It is difficult to analyze how well the map represents the environment because there is no perfect blueprint to compare the map with. However, the update of the static map, where the semi-static objects are removed with the previous dynamic maps seems to work. The evaluation of the static map cannot be presented in any number, but the map seems to represent the environment well, and there are no clear skewed walls etc.

Even if there is no distinct way to evaluate the static map, the map plays a significant role in the localization such that an accurate localization implies that the map is accurate enough. The localization proved to be accurate and therefore the map must reasonably represent the environment well.

Google Cartographer was used for solving the problem with loop-closures, however, the environment where the tests were performed contains no clear loops. The static map creation needs to be tested in the future in more demanding environments to be able to evaluate how well the loop-closures are solved.

6.2 Localization

In Section 5.3 multiple tests were conducted. However, since there is no way of measuring the accuracy of the navigation when using the localization for a trajectory no more conclusions can be drawn other than that the trajectory seems to be correct when compared to the currently used reflector-based system as seen in Figure 5.8 and

Figure 5.9. What however can be analyzed more thoroughly is the initial positioning where the results can be found in Section 5.2 and the achieved repeatability seen in Section 5.5.

6.2.1 Initial positioning

As presented in Section 5.2, the initial positioning algorithms is capable of finding the correct pose of the AGV in around 78% of the time when allowed 2 minutes to estimate the position. As long as an operator is present to restart it when the wrong initial pose is found, this is acceptable. However, in order to replace the current system, this cannot be expected and the initial positioning presented in this thesis must therefore be considered as not robust enough.

There are two main reasons for the failure of the initial positioning, the first one is the time complexity of using more particles. The algorithm tends to have a better chance of finding the correct pose of the AGV given that the number of particles is increased, however, this also increases the time consumption of the algorithm, as presented in Table 5.1. Time complexity was not taken into account when writing the algorithm, however, to give somewhat of a perspective, it would be acceptable for the initial positioning to take up to around 2 minutes in a finished system providing that it finds the correct pose reliably. Another alternative that was not implemented but that might solve many of the problems with the initial positioning is to assume that the AGV has not been moved since the last time it was powered off. With knowledge of the previous pose of the AGV, the algorithm could potentially be reduced to only looking for a correct pose in the nearby area. This could potentially increase the success rate of the algorithm. Another alternative that was discussed during the project was to, for a finished system, including the option to manually drive the AGV to one of many known locations, give the current location as input to the system, and then start the initial positioning with that knowledge, this was however left for future work.

The second problem has to do with the fact that the true state of the AGV might not always be the optimal state according to the evaluation presented in Section 3.5.6. If the AGV is located in a corner with no relevant measurements except for the walls in the corner, it is possible that other corners on the map would provide equal or even better scores. This problem was not tested in the environment however it was noted from the company that this has to be taken into consideration when designing the layout of reflectors. There can not be any two areas that are too similar since that could cause large problems. While it might be less of an issue when using natural features, it is also much harder to solve it if a problem would arise since redesigning the physical environment is not an option.

6.2.2 Repeatability

The correctness of the entire trajectory taken by the AGV is important when planning routes and making sure that no corners are cut which would increase the risk of the AGV stopping due to safety sensors being too close to objects. However, the

most important feature of the localization system is to give the same estimate of the same real position every time regardless of how much time has passed since the initial creation of the map. This is why the results from the repeatability tests are the most important ones in regards to actually being able to deploy the system in a project with high demand for accuracy when handling pallets and so on.

From the online tests where the environment was dynamic but not drastically changed, seen in Figure 5.15, it was found that the accuracy of the system was enough to replace the current system. This might however be dependent on the test scenario and it can not be concluded that this test is sufficient to claim that the proposed localization system will always perform as in Figure 5.15 given similar setups.

When the environment was changed, the reflector-based localization was unable to localize due to the fact that one reflector was constantly blocked. While this was true for the test scenario presented, it is likely that if it was known when constructing the reflector layout that the specific area might be obscured, more reflectors would have been mounted prior to this. This would however have added an additional cost. When it comes to the proposed static and dual-time system, the accuracy for both of the systems would have been sufficient to be used in production with the dual-time system deviating from the assumed mean with a maximum of approximately 8.8 mm in the lateral axis and the static system approximately 10.1 mm. While the shift in the mean of -4.5 mm in the lateral axis for that static system and -4.1 mm for the dual-time system is not a large issue in the presented test. It is worth noting that this was only a single test case and one could assume that at least a similar shift could be found in different directions in other test scenarios. Assuming maximum deviance of around 10mm in any direction, the system would be possible to use in a scenario where pallets are handled and even more so where the demand for accuracy is even lower, such as when transportation from point A to point B is requested without any mm precision required at pickup and drop-off. A relevant real-world example of such a scenario is when transporting bags at an airport. It is also important to note that the measured repeatability is not only based on the localization itself but also the mechanical errors of the AGV.

As for why the difference in the mean is present when the map was changed, no proper conclusions can be made based on tests. However, based on observations during implementation and testing a theory might be worth bringing up. As mentioned in Section 6.1, the actual correcting of the map can not be validated other than that it is accurate enough for localization with accuracy presented in Figure 5.15 and Figure 5.18. The theory from the authors is that the distance between the parallel upper and lower wall on the map in Section 5.17 is not exactly correct but rather a few millimeters too small. This would mean that the optimal y-coordinate would change slightly when seeing both of the walls compared to only one of them. Note again that this is only speculation and not an actual result based on tests.

6.3 Future work

For the map creation, future work could be to not only update the static map by removing objects that are classified as not being static but also add to and expand the map when required. While the initial map creation presented in this work was capable of mapping most of the static objects, a few were not seen during initial mapping and therefore never present in the static map. An option to update the distribution in each cell in the static map might also be preferable as more data regarding the environment is given.

For localization, the project would benefit from being implemented in a way that had time complexity in mind when making design choices such as writing in a language more suitable. The code was for this thesis written in Python 3.8. A way to solve the shift in mean stop position as presented in Figure 5.18 and discussed in Section 6.2.2 would also need attention.

However, the main future work should, according to the authors of this thesis, be the initial positioning algorithm. The algorithm presented is not reliable enough even if the time allowed to find the best pose is increased above the reasonable threshold. This might be mitigated by writing the algorithm in a different, compiled, programming language however it might even be preferable to redesign the algorithm completely. A solution could also be to implement one of the proposed solutions presented in 6.2.1 where information regarding the state of the AGV is manually given as input or the previously known position is assumed to still be relevant. The manually given information might not be an exact pose but instead an area or section of the environment where the AGV is located.

Bibliography

[1] B. Mahadevan and T. T. Narendran, "Design of an automated guided vehicle-based material handling system for a flexible manufacturing system," *International Journal of Production Research*, pp. 1611–1622, 1990. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/00207549008942819

[2] S. Senoo, M. Mino, and S. Funabiki, "Steering control of automated guided vehicle for steering energy saving by fuzzy reasoning," in *Conference Record of the 1992 IEEE Industry Applications Society Annual Meeting*, 1992, pp. 1712–1716.

[3] D. Ronzoni, R. Olmi, C. Secchi, and C. Fantuzzi, "AGV global localization using indistinguishable artificial landmarks," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 287–292.

[4] H. Durrant-Whyte and T. Bailey, "Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms," *IEEE Robotics Automation Magazine*, 2006.

[5] T. Bailey and H. Durrant-Whyte, "Simultaneous Localisation and Mapping (SLAM): Part II State of the Art," *State of the art. IEEE Robotics Autom Mag*, 2006.

[6] K. Krinkin, A. A. Y. Filatov, A. A. Y. Filatov, A. Huletski, and D. Kartashov, "Evaluation of modern laser based indoor SLAM algorithms," in *Conference of Open Innovation Association, FRUCT*, vol. 2018-May. IEEE Computer Society, 9 2018, pp. 101–106.

[7] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D LIDAR SLAM," *IEEE International Conference on Robotics and Automation*, pp. 1271–1278, 6 2016.

[8] G. Grisetti, C. Stachniss, and W. Burgard, "Improving grid-based SLAM with Rao-Blackwellized particle filters by adaptive proposals and selective resampling," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2005, 2005, pp. 2432–2437.

[9] B. Steux and O. El Hamzaoui, "tinySLAM: A SLAM algorithm in less than 200 lines C-language program," in *11th International Conference on Control,*

Automation, Robotics and Vision, ICARCV 2010, 2010, pp. 1975–1979.

[10] P. Biber, "The Normal Distributions Transform: A New Approach to Laser Scan Matching," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 3, 2003, pp. 2743–2748.

[11] T. Kaminade, T. Takubo, Y. Mae, and T. Arai, "The generation of environmental map based on a NDT grid mapping - Proposal of convergence calculation corresponding to high resolution grid," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2008, pp. 1874–1879.

[12] T. Takubo, T. Kaminade, Y. Mae, K. Ohara, and T. Arai, "NDT scan matching method for high resolution grid map," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, 12 2009, pp. 1517–1522.

[13] A. Pålsson and M. Smedberg, "Investigating Simultaneous Localization and Mapping for AGV systems With open-source modules available in ROS," Chalmers University of Technology, Tech. Rep., 2017.

[14] J. Saarinen, H. Andreasson, T. Stoyanov, and A. J. Lilienthal, "Normal distributions transform Monte-Carlo localization (NDT-MCL)," in *IEEE International Conference on Intelligent Robots and Systems*, 2013, pp. 382–389.

[15] R. Valencia, J. Saarinen, H. Andreasson, J. Vallvé, J. Andrade-Cetto, and A. J. Lilienthal, "Localization in highly dynamic environments using dual-timescale NDT-MCL," *IEEE International Conference on Robotics and Automation*, 2014. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6907433/

[16] J. Saarinen, H. Andreasson, T. Stoyanov, J. Ala-Luhtala, and A. J. Lilienthal, "Normal Distributions Transform Occupancy Maps: Application to large-scale online 3D mapping," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2013, pp. 2233–2238.

[17] J. P. Saarinen, H. Andreasson, T. Stoyanov, and A. J. Lilienthal, "3D normal distributions transform occupancy maps: An efficient representation for mapping in dynamic environments," *International Journal of Robotics Research*, vol. 32, no. 14, pp. 1627–1644, 12 2013.

[18] J. Saarinen, T. Stoyanov, H. Andreasson, and A. J. Lilienthal, "Fast 3D mapping in highly dynamic environments using normal distributions transform occupancy maps," in *IEEE International Conference on Intelligent Robots and Systems*, 2013, pp. 4694–4701.

[19] U. Wandinger, *Introduction to Lidar*. Springer, 6 2005. [Online]. Available: https://link.springer.com/chapter/10.1007/0-387-25101-4_1

[20] R. A. Fowler, "The low down on LIDAR," *Earth Observation Magazine*, pp. 1–6, 2000.

[21] A Koubaa, *Robot Operating System (ROS): The Complete Reference*, 1st ed.

Springer Publishing Company, Incorporated, 2016, vol. 1. [Online]. Available: http://www.springer.com/series/7092

[22] S. Riisgaard and M. R. Blas, "SLAM for Dummies A Tutorial Approach to Simultaneous Localization and Mapping," MIT, Tech. Rep.

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

[24] F Cérou and F LeGland, "Efficient particle methods for residual generation in partially observed SDE's," in *Proceedings of the IEEE Conference on Decision and Control*, vol. 2, 2000, pp. 1200–1205.

[25] M. Bolić, P. M. Djurić, and S. Hong, "New resampling algorithms for particle filters," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2, 2003, pp. 589–592.

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY