Integrating the use of prior information into Graph-SLAM with NDT registration for loop detection

Antek Schabert
Örebro University, Sweden
Technology
Integrating the use of prior information into Graph-SLAM with NDT registration for loop detection
Antek Schabert
Örebro University, Sweden

Integrating the use of prior information into Graph-SLAM with NDT registration for loop detection
Integrating the use of prior information into Graph-SLAM with NDT registration for loop detection
Contents

1 Introduction 9
2 Related Work 11
3 Normal Distribution Transform 15
4 Graph-Based SLAM 19
5 Probabilistic Motion Models 21
6 Optimization 23
7 Prior information 27
8 Localization 33
9 Results 35
  9.1 Test setup ............................................................. 35
  9.2 Simulation results .................................................. 37
10 Conclusions 41
Chapter 1

Introduction

With the increased use of autonomous agents in open and often changing environments it is important that these agents have the ability to accurately localize themselves in their environment and build a map of the environment. If global localization systems such as GPS are not available or are not accurate enough, agents often have to rely on range sensors and odometry information for localization. Building a map and at the same time using the map for localization is known as simultaneous localization and mapping (SLAM) and is one of the most important problems in robotics. It is necessary for an efficient solution to many other problems, such as path planning and multi-agent coordination. Most SLAM approaches maintain an uncertainty estimation of the position of the agent and try to detect features in the map that were detected previously to reduce this uncertainty and improve the position estimation and, therefore, also the map quality.

Although several techniques for SLAM exist, they are still not mature enough for use on heavy production vehicles. As an effort to improve reliability of a mapping and localization system, one direction of research is the use of prior information. If a robot system could exploit prior information - in the form of, e.g., aerial or satellite images, CAD drawings, or hand-drawn sketches - it should be able to build a detailed local map using its own sensors more quickly and more accurately. However, matching information from such a rough map (which may be out of date, or using different features compared to what the robot can directly observe) to the robots observations is a non-trivial task.

In this paper an algorithm will be presented that combines Graph-SLAM with pose estimates extracted from running MCL on a prior map. However, unlike in other papers only radar sensors will be used as input. Since radar sensors are allowed for use on public streets in most parts of the world this makes the results of this paper widely useable. On the other hand radar sensors generally provide less data from the environment and more noise than 2D laser range finders. To handle this noise and minimize the error introduced by it NDT (Normal Distribution Transform) scan registration will be used for loop detection.
Chapter 2
Related Work

While the research focus on SLAM algorithms has generally improved the quality of the generated maps in comparison to the results of widely used algorithms from 20 years ago. But at the same time the diversity of possible algorithms with their own strengths and drawbacks makes the decision which algorithm to use more complicated. This is in our case even more so, since most algorithms are defined without additional information sources in mind and the work to extend the algorithms to accept such input has to be estimated as well as the type of information that could be used. In this chapter we will give an overview of the currently common SLAM algorithms and give a quick opinion about how additional information may be introduced to the algorithms. Since the SLAM problem is closely related to the localization problem it is not surprising that many approaches to the problem follow a similar idea as the localization algorithms Monte Carlo Localization and the Extended Kalman Filter.

Particle based algorithms similar to the one presented by Montemerlo et al. [1] generate a multi-hypothesis system where the agent draws a different sample from the motion model for each hypothesis and builds a map for each hypothesis as well. This works well if the agent often moves back into already mapped areas where inaccurate particles can be discarded. While prior information could be used to run the localization step on the prior map as well and particles could be discarded according to a mixture of the weight of the localization on the prior map and the local map, the best particle might be discarded to fit outdated information and it would be impossible to recover this information once we reach an area that was already mapped again.

On the other hand solutions to the SLAM problem that use the extended Kalman filter [2] or the sparse extended information filter [3] for the localization make strong assumptions about the motion model and the sensor noise. If these assumptions do not represent the reality, the errors from the approximation will accumulate and eventually lead to inconsistencies [4]. If the positions of landmarks were given on a prior map or could be calculated these known locations could be used as an initial estimate for the landmarks. This would directly improve the uncertainty of the filter when a landmark is detected for the first time.

In contrast Lu and Milios proposed to keep all scans of the environment together with relations between different poses and the covariance matrix of those relations in memory [5]. The relations can, for example, be derived from the odometry or scan matching. Finally, they propose to use least squares optimization to find the most likely poses of the agent. This representation is now commonly known as Graph-Based SLAM or Graph-SLAM. While this allows to correct errors made in the mapping, the computational complexity being $O(n^3)$ in relation to the poses $n$ made this approach only practical for problems with few poses. However, since then, more efficient algorithms for the optimization problem were presented and are now easily available [6]. This allows the use of Graph-Based SLAM also on problems with many more poses. Prior information can easily be introduced as fixed points in relation to an agent pose.
CHAPTER 2. RELATED WORK

The uncertainty of the prior information is then used for the covariance or information matrix of the relation. Aside from particle based approaches that only measure the likelihood of a scan, given the map of the particle, all these algorithms rely either on landmark or loop detection to reduce the error in the estimated pose of the agent. In both cases the core of the problem is often to match two scans to find the relative pose between the positions where the scans were taken. This is known as scan registration and can be divided into two categories - local search strategies and global search strategies. Local search strategies require a good initial guess of the relative pose and easily get stuck in local minima if the guess is too far off from the correct pose.

A straightforward approach in this category is to find for every point in one scan the closest point in the other scan and minimize the distance as proposed by Besl [7]. This is commonly known as the iterative closest point (ICP) algorithm and is probably the most widely used solution to scan registration. The biggest drawback of it is that the search for the closest point is quite time consuming. While this problem can be reduced by storing the points in a semi-organized form and sampling in dense areas, a different representations of the point clouds can avoid it altogether.

In the Normal Distribution Transform (NDT) [8] a point cloud is represented by a collection of local normal distributions, where each normal distribution represents the likelihood distribution of a point falling into a certain area. These normal distributions are typically ordered in a grid or an N-tree structure, so that finding the closest normal distribution for a point is a simple lookup operation or search of the neighboring cells if the initial cell was empty, instead of iterating over a whole point cloud. In addition, normal distributions have analytic derivatives, which allows the use of standard optimization methods for the registration process.

Another approach from Stoyanov et al. converts both, the reference and the registration point cloud, into an NDT representation and minimizes the L2 distance between the NDT maps [9]. While this leads in many cases to faster computation times unlike regular NDT registration this approach has, assuming that both NDT maps have the same amount of cells, a quadratic computational complexity in regard to the number of cells of the NDT maps. This makes the choice of the cell size a more important factor and works better for the registration of two independent scans rather than registration of a scan to a map.

On the other hand global search strategies do not rely on an initial guess, but rather search over all possible transformations. Since this is very time expensive, strategies often include additional discretization [10] or using local feature descriptors that are rotation invariant [11] to reduce the search space.

Including information from prior maps into the SLAM process is a rather new research area and there are two important factors that have to be considered. First we can not assume that the map is accurate, since it was most likely taken at a different time and it might lack details depending on how it was created. Second the map might include different information depending on the sensor that was used to create the map and the vantage point of the sensor, or if the map was created by a sensor at all. Kümmerle et al. show how localization on prior information can be used to improve the accuracy of a SLAM algorithm [12], but does not consider inaccuracies of the map. Closely related Vysotska and Stachniss [13] use Open Street Maps to bound the error of Graph-SLAM, but use ICP registration instead of MCL to extract the information from the prior map. Parsley et al. propose to use prior information as bounds for the location of landmarks instead of influencing the pose of the agent directly [14]. While all these publications use laser range finders we will be using radar sensors, which are already used in many modern vehicles for parking assistants, collision avoidance and other driving assistance systems. In contrast to laser range finders radars have a lot less valid detections, may detect occluded objects, don’t have a fixed angle difference between detections and more noise than laser range finders.
On the other hand, including prior information about the structures in the environment can especially improve rotational uncertainties. For example, Newman et al. proposed to search for line features that are almost parallel or orthogonal and add artificial observations to make these features fully parallel or orthogonal [15].

In his doctoral work Parsley also discussed the integration of different information of observed features, but the used information is generated by hand rather than from a map [16].
Chapter 3
Normal Distribution Transform

Scan registration is the process of finding an affine transformation function that projects the sensor readings of one time-frame onto the sensor readings of another time-frame and minimizing the difference between the two scenes. It is used in various areas in robotics and also other research fields. For example, it can be used to provide accurate estimations of the odometry, recognize loops in SLAM algorithms and detect moving objects. The result of a NDT scan registration can be seen in Figure 3.1.

![Figure 3.1: Results of a NDT scan registration. Black is the reference map, red is the initial guess of the scan and blue the result after scan registration.](image)

However, sensor readings are often acquired in the form of point clouds which contain a lot of redundant information - the points are mostly samples from surfaces the sensor detected. While most registration algorithms work directly on these point clouds, this is a computationally heavy process and does not take advantage of any information about the surface itself. Biber and Strasser [8] proposed to divide two-dimensional point clouds into cells and calculate a normal distribution for every cell. These normal distributions now approximate the local surface, from which the points...
were sampled, and provide smooth gradients, the orientation of the surface, as well as the probability of a point being sampled from the corresponding normal distribution in a certain position. While the whole map now is represented as a Gaussian mixture model of the normal distributions, the normal distribution of a cell alone provides a good enough local approximation over the area of the cell and the error of calculating the probability of a point falling into the cell without using the normal distributions of the rest of the map is insignificant. This can be assumed since all normal distributions were calculated using only points inside the cell and, therefore, have their mean inside the cell and a covariance matrix that does not overlap much with other cells.

In more detail the equation 3.1 now provides for every point \( x \) an easy way to calculate the probability that the point was generated by the same structure that generated the points used for the normal distribution.

\[
p(x) = \frac{1}{(2\pi)^{D/2} \sqrt{\det{\Sigma}}} \exp\left(-\frac{(x - \mu)^T \Sigma^{-1} (x - \mu)}{2}\right)
\]  

(3.1)

with the mean

\[
\mu = \frac{1}{n} \sum_{k=1}^{n} x_k,
\]

(3.2)

the covariance

\[
\Sigma = \frac{1}{n-1} \sum_{k=1}^{n} (x_k - \mu)(x_k - \mu)^T
\]

(3.3)

and the dimension \( D \).

With a probability for each point we can now define the optimal transformation of the point cloud as the one that maximizes the product of the probabilities of the points [17].

\[
\psi = \prod_{k=1}^{n} p(T(p, x_k))
\]

(3.4)

where \( T(p, x_k) \) is the transformation function applied to the point \( x_k \).

However, with this equation single outliers would have a strong impact on the resulting transformation. To avoid such a behavior it is common to use a mixture of a normal distribution and a uniform distribution.

\[
\bar{p}(x) = c_1 \exp\left(-\frac{(x - \mu)^T \Sigma^{-1} (x - \mu)}{2}\right) + c_2 p_o
\]

(3.5)

where the constants \( c_1 \) and \( c_2 \) can be calculated by applying the requirement that the integral of the normal distribution over the cell has to be one and \( p_o \) is the expected rate of outliers.

It is also common in optimization problems to minimize a score function instead of maximizing it. Also, by taking the logarithm of the probabilities the product can be turned into a sum. This way, the derivative of the resulting function

\[
-\log \psi = -\sum_{k=1}^{n} \log p(T(p, x_k))
\]

(3.6)

can now be calculated for each summand independently. While this score function is robust to outliers the derivatives of a logarithm are typically not fully defined. Biber also proposed to approximate the logarithm of a probability with a new Gaussian function [8].
\[ d_3 = -\log(c_2) \]
\[ d_1 = -\log(c_1 + c_2) - d_3 \]
\[ d_2 = -2\log((-\log(c_1 \exp(-1/2) + c_2) - d_3)/d_1) \]

\[ \hat{p}(x) = -d_1 \exp\left(-\frac{d_2}{2}(x - \mu)^T \Sigma^{-1}(x - \mu)\right) \]

This approximation provides the same characteristics for optimization but has simpler derivatives if used for the score function

\[ s(p) = -\sum_{k=1}^{n} \hat{p}(T(p, x_k)). \tag{3.9} \]

Newton’s algorithm is a well known optimization algorithm that iteratively solves the equation

\[ H \Delta p = -g \tag{3.10} \]

where \( H \) is the Hessian matrix and \( g \) is the gradient vector of \( s(p) \). The result \( \Delta p \) is then added after each step to the transformation vector \( p \). The gradient vector and the Hessian matrix can be derived by calculating the first and second order partial derivatives of \( s(p) \) respective to the coefficients of \( p \). The entries of the gradient vector are

\[ g_i = \frac{\delta s}{\delta p_i} = \sum_{k=1}^{n} d_1 d_2 x_k^T \Sigma_k^{-1} \frac{\delta x'_k}{\delta p_i} \exp\left(-\frac{d_2}{2} x_k^T \Sigma_k^{-1} x_k'\right) \tag{3.11} \]

where \( x'_k = T(p, x_k) - \mu_k \). \( \mu \) as well as \( \Sigma \) now have subscripts to indicate that they are taken from the cell the point \( x_k \) falls into after being transformed by \( p \). In a similar fashion the equation for the entries of the Hessian matrix are

\[
H_{ij} = \frac{\delta^2 s}{\delta p_i \delta p_j} = \sum_{k=1}^{n} d_1 d_2 \exp\left(-\frac{d_2}{2} x_k^T \Sigma_k^{-1} x_k'\right) \\
\left(-d_2 \left(x_k^T \Sigma_k^{-1} \frac{\delta x'_k}{\delta p_i}\right) \left(x_k^T \Sigma_k^{-1} \frac{\delta x'_k}{\delta p_j}\right) + x_k^T \Sigma_k^{-1} \frac{\delta^2 x'_k}{\delta p_i \delta p_j} + \left(\frac{\delta x'_k}{\delta p_j}\right)^T \Sigma_k^{-1} \frac{\delta x'_k}{\delta p_i}\right). \tag{3.12}
\]

While these equations may look complex the computational complexity is \( O(n + m) \) with \( n \) being the number of points in the scan to be matched and \( m \) being the number of points in the scan used for the normal distributions. This is easy to see since all equations used here are scaling linearly with the number of points. The only other task we need to perform is to pool the points of the point cloud used for the NDT cells into their respective cells. Since the corresponding cells can be found by a simple look-up action this only takes linear time as well if a data structure with linear computational complexity is used to store the points for each cell.

In comparison ICP has a complexity of \( O(n \times m) \). While the idea behind ICP is quite simple, the algorithm has to search for each point in the point cloud for its nearest neighbor in the fixed point cloud. Without optimization this has a computational complexity of \( O(n \times m) \). While this runtime can be reduced in average, by pooling the points before the matching, the worst case still stays \( O(n \times m) \).

This makes NDT scan registration on dense point clouds a lot faster. On the other hand it can also improve the accuracy of the result if the used sensors are subject to a lot of noise.

We will be using NDT scan registration in this work. While our sensor data is not very dense our sensors are subject to a lot of noise. While ICP would also work,
the results would be influenced more by the initial guess. For example if we tried to register 2 scans of a wall ICP would probably, depending on which side of the wall the initial guess started, end a bit away from the actual mean to the side it started from.
Chapter 4
Graph-Based SLAM

One of the central problems in robotics is the building of a map of an unknown environment by only using the data of on-board sensors. This problem is known as simultaneous localization and mapping or in short SLAM. This process is complicated since the measurements of sensors are subject to many noise sources. Without correcting these errors the uncertainty of the real pose of the robot will increase over time and make a consistent alignment of the range measurements impossible.

Many different approaches to this problem were published in the past. Most of them calculate the current pose from the last pose and the odometry readings directly and keep track of the uncertainty of this pose in another way. When additional information, e.g. through loop or landmark detection, the current pose is recalculated from the likelihood fields of the new information and the uncertainty around the pose. However in most cases neither old poses nor the map built from these old poses is corrected.

Graph-Based SLAM on the other hand is keeping all scans, poses, relative information between poses and the uncertainty of the relative information in memory and uses optimization algorithms to correct not only the current pose but also all past poses of the robot. The representation used here can be described as a directed graph. The best estimations for the poses of the robot are stored in the nodes. The edges in the graph on the other side store a probability distribution of the relative movement or observation from one pose to another. This information is generally represented as a vector of the special Euclidean group $\mathbb{SE}(2)$ and a covariance or information matrix of a normal distribution. The mean value of the normal distribution is not stored separately since it is considered to be the movement or observation vector. When the graph is build the edges are, for example, set to the odometry readings and the probability distribution according to the motion model. The initial pose of the nodes is then determined by assuming that the robot performed the movement stored in the edge from the pose of the last node in the graph. However, once enough observations are added to the graph these poses will iteratively be changed by minimizing an error function of the edges.

To perform such an optimization it is necessary to have a fixed anchor point in the graph and additional edges aside from the odometry readings. The anchor point is generally assumed to be the first node, which will not be optimized. While the actual pose of this node is not relevant, it is generally assumed to be the origin of the coordinate system.

Additional edges are typically generated by performing scan registration or landmark detection. Figure 4.1 shows how a Graph with such information may look like.

Scan registration is typically performed by matching the current scan to scans of other poses that are assumed to be close enough to provide a good initial guess. The resulting transformation vector is then added as a new edge to the graph between the two scans. Calculating a plausible information matrix for the edge on the other hand is
a more complicated problem and depends on the scan registration algorithm that was used. If ICP was used a covariance matrix may be calculated from the points of the current scan and the corresponding closest neighbor points. The information matrix is then simply the inverse of the covariance matrix. If NDT scan registration was used the Hessian matrix can be used as an estimate for the information matrix. However, in most cases these matrices are inaccurate estimations of the true error. Especially if the algorithm finds a local minimum the information matrix will not represent the error caused by this.

Landmark detection on the other hand extracts features from the current scan and adds a new edge between the current pose and a node that represents the feature. Similar to scan registration the information matrix of the edges is considered to represent the error from the sensors and the algorithm used to detect the feature. False detection of landmarks will generally lead to large inaccuracies in the map. Landmark detection will not be used in the algorithm presented in this paper, since the main focus of this work is to include prior information into Graph-SLAM. While prior information alone would be insufficient for good results, loop detection was simpler to implement and already provides good results. Landmark detection on the other hand would most likely not find unique features with the sparse data from the radar sensors and make matching the feature depending on the estimated pose necessary. How landmark detection would be used in Graph-SLAM is mainly included here for completeness.

In scan registration as well as landmark detection the problem of false positives can be seen as a simple outlier problem. During optimization a false constraint will add a large error to the nodes it is connected to and therefore change the pose of these nodes. This error will then propagate throughout the whole graph. While the common way to handle outliers would be to use a probability distribution model that limits the influence a single point can have on the complete probability we have to consider that the graph used for Graph-SLAM is in most cases sparse. This means the nodes have very few constraints and even with a different probability model a wrong constraint could still have a strong negative impact during optimization. Instead the common approaches to this problem are to find the constraints, which add an unproportional large amount to the total error of the graph and use switchable constraints [18], which may be deactivated by the optimization algorithm or to allow the optimization algorithm to scale the covariance of outliers to reduce their influence on the graph [19].
Chapter 5
Probabilistic Motion Models

In the last chapter we introduced that the edges of the graph in Graph-SLAM store the odometry readings and the corresponding information matrix. However, to provide a plausible information matrix we need to have a model of the motion and the noise of the sensors measuring the motion.

The most common model was described by Thrun et al. in [21] and describes the motion of the robot over a short time frame as a rotation before the motion, a straight motion and a rotation after the motion. Figure 5.1 shows how such a motion looks like. These three components are then considered to be subject to noise from the sensors.

\[
\begin{align*}
\hat{\delta}_{\text{trans}} &= \delta_{\text{trans}} + \varepsilon_{\text{trans}} \quad \varepsilon_{\text{trans}} \sim \mathcal{N}(0, \sigma_{\text{trans}}^2) \\
\hat{\delta}_{\text{rot1}} &= \delta_{\text{rot1}} + \varepsilon_{\text{rot1}} \quad \varepsilon_{\text{rot1}} \sim \mathcal{N}(0, \sigma_{\text{rot1}}^2) \\
\hat{\delta}_{\text{rot2}} &= \delta_{\text{rot2}} + \varepsilon_{\text{rot2}} \quad \varepsilon_{\text{rot2}} \sim \mathcal{N}(0, \sigma_{\text{rot2}}^2)
\end{align*}
\]  

(5.1)

The error of the motion is modeled as normal distributions around these three components of the motion. However, the standard deviations of the normal distributions depend on the length of the movement and the absolute of the rotations.

\[
\begin{align*}
\sigma_{\text{rot1}} &= \alpha_1 |\delta_{\text{rot1}}| + \alpha_2 \delta_{\text{trans}} \\
\sigma_{\text{trans}} &= \alpha_3 \delta_{\text{trans}} + \alpha_4 (|\delta_{\text{rot1}}| + |\delta_{\text{rot2}}|) \\
\sigma_{\text{rot2}} &= \alpha_1 |\delta_{\text{rot2}}| + \alpha_2 \delta_{\text{trans}}
\end{align*}
\]  

(5.2)
α₁ to α₄ are the parameters that have to be tuned to fit the accuracy of the sensors measuring the motion.

Due to the noise of the rotation before the motion the probability distribution after the motion takes on a bent form if projected to a plane only showing the position without the angle and it is not possible to describe it with a normal distribution in general.

However, if the noise to the rotation is rather low, a normal distribution provides a good approximation. In this case the Gaussian probabilistic motion model may be used. Here the motion is represented as a typical vector of the special Euclidean group 2 or in short as a SE2 vector. The components of this vector are a forward part of the motion, a lateral part of the motion and a rotation after the motion. Once again the noise is added onto these components

\[
\hat{\Delta}^{odo}_x = \Delta^{odo}_x + \epsilon^{odo}_x \sim \mathcal{N}(0, \sigma^2_{\Delta^{odo}_x}) \\
\hat{\Delta}^{odo}_y = \Delta^{odo}_y + \epsilon^{odo}_y \sim \mathcal{N}(0, \sigma^2_{\Delta^{odo}_y}) \\
\hat{\Delta}^{odo}_\phi = \Delta^{odo}_\phi + \epsilon^{odo}_\phi \sim \mathcal{N}(0, \sigma^2_{\Delta^{odo}_\phi})
\]  

(5.3)

and the noise depends on the length of the motion and the rotation.

\[
\sigma_{\Delta^{odo}_x} = \sigma_{\Delta^{odo}_y} = \sigma^{\min}_{xy} + \alpha_1 \sqrt{(\Delta^{odo}_x)^2 + (\Delta^{odo}_y)^2 + \alpha_2 |\Delta^{odo}_\phi|} \\
\sigma_{\Delta^{odo}_\phi} = \sigma^{\min}_{\phi} + \alpha_3 \sqrt{(\Delta^{odo}_x)^2 + (\Delta^{odo}_y)^2 + \alpha_4 |\Delta^{odo}_\phi|}
\]  

(5.4)

While this model is theoretically less accurate, in practical use the rotation after the motion may also represent the rotation before the motion of the next node in Graph-SLAM. This leaves the first edge without a rotation before the motion, but in Graph-SLAM the first pose is in general an arbitrary pose, which is fixed and simply represents the beginning of the map. As such, unrepresented errors in the first pose do not affect our algorithm and can be ignored.

On the other hand this model is easier to integrate into Graph-SLAM and also provides a better way to model position errors from other effects than rotational uncertainty. We will be using the Gaussian probabilistic motion model in this work. However, we assume both motion models would provide a viable foundation for Graph-SLAM and later on localization on the prior map.

---

**Figure 5.2:** Gaussian probabilistic motion model [20]
In the chapter about Graph-SLAM we touched the topic of optimizing the graph as part of the algorithm. Since we already explained how Newton’s algorithm can be used to optimize the score function during scan registration with NDT maps one might expect that the score function of the graph may be minimized in the same manner. $z_{ij}$ represents the mean of the constraint between the nodes $i$ and $j$, $\Omega$ represents the information matrix and the vectors $x$ represent the pose variables of the nodes. The Function $e$ is a vector error function that measures how well the parameters of the nodes satisfy the constraint. $C$ simply represents the node pairs of all edges in the graph.

For a shorter notation we can define

$$e(x_i, x_j, z_{ij}) = e_{ij}(x).$$

The idea of the Gauss-Newton method is to approximate the error function by its first order Taylor expansion,

$$e_{ij}(x + \Delta x) \approx e_{ij} + J_{ij} \Delta x,$$

where $J_{ij}$ is the Jacobian of $e_{ij}$ computed in $x$. Substituting the terms of equation 6.1 we now get

$$F_{ij}(x + \Delta x) \approx (e_{ij} + J_{ij} \Delta x)^T \Omega_{ij} (e_{ij} + J_{ij} \Delta x)$$

$$= e_{ij}^T \Omega_{ij} e_{ij} + 2 e_{ij}^T \Omega_{ij} J_{ij} \Delta x + \Delta x^T J_{ij}^T \Omega_{ij} J_{ij} \Delta x$$

$$= c_{ij} + 2 b_{ij} \Delta x + \Delta x^T H_{ij} \Delta x$$

and

$$F(x + \Delta x) \approx c + 2 b \Delta x + \Delta x^T H \Delta x$$

with

$$c = \sum c_{ij},$$

$$b = \sum b_{ij}$$

and
H = \sum H_{ij}. \tag{6.8}

This equation may then be minimized by setting the Jacobian of this approximation to 0 and solving the resulting linear system

H\Delta x = -b. \tag{6.9}

The factor 2 was omitted here since the result \Delta x would then be multiplied with a scalar before adding it to x anyways.

However, the Gauss-Newton method is a general approach to multivariate function minimization. It works under the assumption that the space of the variables is Euclidean. While the position variables in Graph-SLAM clearly form an Euclidean space, the orientation of the robot does not.

Using the Gauss-Newton method with variables that span over a non-Euclidean space is a problem because these variables could cause singularities. To avoid singularities it is common to represent the rotation in an over-parametrized way, e.g., by rotation matrices or quaternions. However, if the Gauss-Newton method is directly applied to these representations the update step might break the constraints induced by the over-parametrization. On the other hand if a minimal representation like Euler angles is used the algorithm is subject to singularities [6].

A common approach to handle variables that span over non-Euclidean spaces is to choose different representations for the variable x and the incremental variable \Delta x [6]. Since the values of \Delta x are typically small and far from the singularities, it is safe to use a minimal representation here. On the other hand x may be represented in an over-parametrized way to avoid singularities.

For the Gauss-Newton method to work with different representations for these variables we need a new update function \oplus that adds \Delta x to x.

\[ x' = x \oplus \Delta x \tag{6.10} \]

This new function would also have to be used to calculate the analytical Jacobian. However, it is also possible to approximate the Jacobian numerically.

Another possible solution would be to use a minimal representation for both variables and resolve the singularities whenever necessary. The poses of a 2D Graph-SLAM algorithm might, e.g., be represented as the translation and an Euler angle. While a pose may only have a rotation between 0° and 360° the optimization may as well lead to angles outside of this scope. However, since we know that rotation functions repeat themselves after every 360° the angle can be mapped to a valid value again.

If we take another look at equation 6.8 we can see that the matrix H is simply summed up over all edges. Moreover, each term only depends on the variables of the connected nodes. Therefore, all derivatives with respect to variables that do not appear in the term will be 0 and the Jacobian will have the form

\[ J_{ij} = \begin{pmatrix} 0 \ldots 0 & A_{ij} & 0 \ldots 0 & B_{ij} & 0 \ldots 0 \end{pmatrix} \] \tag{6.11}

where \( A_{ij} \) represent the derivatives of the error function with respect to \( \Delta x \), and \( B_{ij} \) represent the derivatives of the error function with respect to \( \Delta y \). With the Jacobian we can now calculate \( H \) and \( b \) as shown in equation 6.4:

\[ H_{ij} = \begin{pmatrix} \cdots & A_{ij}^T \Omega_{ij} A_{ij} & \cdots & A_{ij}^T \Omega_{ij} B_{ij} & \cdots \\ B_{ij}^T \Omega_{ij} A_{ij} & \cdots & B_{ij}^T \Omega_{ij} B_{ij} & \cdots \\ \end{pmatrix} \]

\[ b_{ij} = \begin{pmatrix} \cdots & A_{ij}^T \Omega_{ij} e_{ij} & \cdots \\ B_{ij}^T \Omega_{ij} e_{ij} & \cdots \\ \end{pmatrix} \] \tag{6.12}
where the dots represent 0 values.

It can be seen that the entries of the matrix $H_{ij}$ are only in the blocks of the variables of the two nodes connected by the edge. In fact the resulting matrix $H$ forms the adjacency matrix of the graph, if we just check if a block contains non-zero values. Since the graph of a typical SLAM run is very sparse the majority of the entries of the matrix $H$ will contain zero values. With such a sparse matrix, solving the linear system in equation 6.9 may be accelerated by using specialized solvers [22]. Another characteristic of the matrix is that it is symmetric and positive-definite which makes it possible to solve the linear system with the Cholesky decomposition which can also be optimized for sparse matrices [23]. We will be using the sparse Cholesky decomposition in our algorithm, because we will be adding additional edges to the graph from our prior information. If the density of the matrix increases the runtime of the Cholesky decomposition is more stable than other methods.

On the other hand Rosen et al. recently published a solver that does not rely on the Gauss-Newton method at all and provides a global solution to the optimization problem [24]. While we would have liked to try this method there was no public available implementation in C or C++ during the implementation phase of this work.
Now that we have a SLAM approach that can be extended to use additional information without problems, it is necessary to consider what type of errors and information different maps could contain and if they could be used to improve the quality of the SLAM algorithm.

Parsley introduced a distinction of four types of information in prior maps [25].

The first type is absolute information. In this case it is possible to extract the pose of the robot or the position of a feature on the prior map and project this to a pose or position in the map the robot is building. These poses can be extracted by running a localization algorithm on the prior map or calculating the position of features on the prior map and then detecting these features during the SLAM algorithm.

This type of information would obviously benefit the SLAM algorithm greatly. In fact, if we always had access to the exact pose of the robot the SLAM problem would be solved. Unfortunately, the poses extracted from a prior map are subject to different error sources. If the pose was the result of a localization algorithm the algorithm itself already provides a covariance matrix for the error. However, this error estimation would assume that the map contains no errors in itself. But in most cases the majority of the pose error will be due to inaccuracies of the prior map and these errors are harder to calculate. Nonetheless, even if it is not feasible to get an accurate error estimation for these errors, if we can provide an upper bound of these errors, the resulting poses will also bound the error of the SLAM algorithm. While this boundary on the error may not be enough to provide an acceptable map it can improve the success rate of the loop detection and, therefore, indirectly improve the map.

One problem with using absolute information is to get a correct transformation function. For most type of maps this function will be a rigid transformation and it is only necessary to have an accurate estimation of the starting position on the prior map. It is also possible that the map has a different scale, but this can be changed in a simple preprocessing step.

The second type of information is relative information. While the position of landmarks is not known in this case, it is possible to find a correlation between these landmarks. The most common relation exploited is parallelism and orthogonality in urban and indoor environments, but it could also include e.g. the distance between beacons. While most approaches using parallelism and orthogonality enforce these constraints on the whole map, assuming that the algorithm is run in a urban or indoor environment, extracting this kind of information from a map for single walls or corners would be possible as well. However, to improve the SLAM algorithm it would be necessary to match walls and corners from the prior map to the map build during SLAM and enforce the relative information there.

The third type is topological information. Topological information view the world as a collection of places that are connected. The robot can then navigate by following the connections between places without the need for accurately mapping the envi-
CHAPTER 7. PRIOR INFORMATION

This is possible in environments with easy to recognize transitions between places like doors in indoor environments. While this kind of information is unlikely to be present in maps, having a topological overview would reduce the loop detection problem to detecting the transition between topological points. Since transitions between topological points are often easy to detect features like doors this would make loop detection easier.

The last category is semantic information. This consists of additional information about features retrieved through computer vision techniques from camera images, which makes it possible to reason about the likelihood of the feature being true or false. While many prior maps are recorded by cameras from a high point of view, extracting this kind of information would exceed the scope of this work.

On the other hand the most common errors present in maps can be categorized into systematic errors, lack of detail and the map being out of date.

Systematic errors, like a difference in the scale of the maps, and known distortions of the image can be corrected in a preprocessing step. Therefore, it is possible to assume for this work, without loss of generality, that the maps will contain no such errors.

A lack of detail can include that surfaces which are close to a geometrically simple shape are approximated in the map as such a shape, e.g. building walls may be approximated by a line. But it also includes that minor objects in the area may have been omitted. These errors would only have minor effects on a localization algorithm but would likely cause problems with feature detection and if relative information were extracted from the map.

Harder to account for is, if the map is out of date. While it may be possible to assume upper bounds in areas with known dynamic changes, like parking spaces, if the map went through major changes using it as prior information may have a negative impact on the SLAM algorithm. Unfortunately, it is hard to say beforehand if the changes in a map will cause problems or not.

At last it also could be considered as an error that maps are generally only covering a limited area. However, this is unlikely to have a negative impact on the SLAM algorithm since outside of the known area there would simply be no new information provided from the prior map.

Figure 7.1: Building Plan
Taking a look at the literature on using prior maps the most common types of maps are hand-drawn maps, satellite or aerial images and blueprint-like maps. The focus, when hand-drawn maps are used, is typically not to improve the SLAM algorithm but rather to find a middle ground between the hand-drawn map and the map the robot is building with the goal of later using additional information like directions present on the hand-drawn map [26][27]. Recognizing such information is useful for robots that have to interact with humans, but can be considered uninteresting for heavy production vehicles, which are the target of this work.

Satellite or aerial images will most likely lack fine details and include details and information which can not be seen by the robot, but will be accurate aside from this. However, the information in these maps is not directly accessible. Rather, it is necessary to extract the required information using image processing techniques. Depending on how accurate this extraction works even more details will be lost and the result can be considered similar to a blueprint-like map. However, since image processing techniques may also detect false or miss relevant details this extraction is often done by hand for research purposes [12].

Blueprint-like maps will typically lack a lot of details of the real area but will still provide strong boundaries for a localization algorithm. The most likely features that could be extracted from such a map would be corners and lines, but the scans of the robot would detect more corners and less constant lines because of the additional details it can recognize. While detecting such features would be possible, matching a detected feature to the correct feature on the prior map would be prone to errors. To minimize the effect of such errors it would be necessary to use switchable constraints in the Graph-SLAM algorithm.

Figure 7.1 shows a building plan we will be using in this work. Aside from the buildings we can see additional information in a gray font. Mainly the building numbers and the outlines of the streets and other borders that are easy to recognize for humans like fences. Since many of these structures will not be visible on a range scan the information displayed in gray can be filtered out. It can also be seen that the scale of the map is 1:1500. Since the map was meant to be printed as an A3 document, this information allows us to convert the map to a correct scaled grid map. The building outlines contain, as expected from such a plan, very little details. A typical range scan...
will pick up other details of the buildings, e.g. entries and railings, not contained in this map.

**Figure 7.3:** Building Plan and scans overlayed

**Figure 7.4:** Closer view of the overlayed area

In more detail, in Figure 7.2 we can see the scans of the agent we use in this paper with accurate pose estimates from a RTK GPS. Without additional information it is hard to tell which part of the map was explored. But with an accurate guess for the
initial pose we can project the scan onto the map and see the differences in detail in Figure 7.3 and 7.4.

We can see that the truck gathers accurate information about most walls present in the map when it drives by. But there is also a lot of additional information in the scans and even some differences between the map and the scans. The building in the middle to the left, for example, is actually a gas station and the map shows the area that the roof covers while the agent sees the gasoline pumps below the roof. The building in the middle to the right seems to be bigger on the scans. This is because next to the building there is a container and some parking cars. As we mentioned, such differences would make feature detection without switchable constraints an error prone strategy.

To work reliably with most maps this work will focus on extracting absolute information through localization.
Chapter 8
Localization

To extract absolute information from the prior map through localization there are multiple approaches we could use. Two common algorithms to the localization problem are the extended Kalman filter (EKF) and the Monte Carlo Localization (MCL). However, since we want to localize on an inaccurate map, our requirements to a localization algorithm are slightly different. Since the prior map may contain blank areas, where the uncertainty of the localization algorithm could quickly increase, a localization algorithm that supports multi-hypothesis upon reentering known areas is preferable. Since the EKF does not support multi-hypothesis and other papers already used MCL successfully on prior maps [12] we decided to use MCL in this paper as well. While Vysotska and Stachniss used ICP registration successfully for the localization problem on a prior map it is unclear how well this behaves on incomplete maps where multiple hypotheses might be necessary.

The idea behind MCL is quite simple. Instead of representing the pose of the robot as a probability function it is represented by multiple particles which contain pose information where the robot could be. If the robot moves, the particles each draw a sample from the motion model of the robot and use this noisy motion instead of directly using the odometry information. Since every particle draws a different sample the particles then spread out. This step is known as the motion update.

The other step of MCL is the sensor update. This step is triggered, if we have a range scan of the robot at the current pose, and calculates the probability of each particle perceiving the current range scan from its current pose. With these probabilities the unlikely poses are then thinned out and new copies of the likely poses are created. This step is commonly known as resampling. However, unlike with the motion update there are multiple approaches to the calculation of the probabilities as well as the resampling.

The most accurate algorithm for the probability calculation of the sensor update is to trace the rays of each detection from the sensor to the detected point. This way detections of objects with other obstacles in the path of the ray can also be considered to be unlikely. However, this process is rather slow and prior maps will rarely have enough details in front of walls to really make use of this feature. It also has a unsmooth probability distribution over the map and would lead to large errors if an object present in the map would not appear on the sensor readings. A considerably faster solution was presented by Thrun et al. [21], which only uses the detected points without tracing the path to the detection. Since in this case the pose of the sensors is only relevant to transform the sensor readings, it is possible to calculate a lookup table in the size of the map that contains the likelihood of a detection being seen at a certain point in the map. While calculating such a likelihood field may be slow, depending on the map, it only needs to be calculated once and can then be used as long as the map is not changed.
Each cell in the lookup table can be computed as a function of the distance from the cell $c_{ij}$ to the closest occupied cell $o_{ij}$ on the map $m$

$$p(c_{ij}|m) = z_{\text{random}} + z_{\text{hit}} \times \frac{1}{\sqrt{2\pi\sigma^2_{\text{hit}}}} \exp \left( -\frac{c_{ij}^2}{2\sigma^2_{\text{hit}}} \right), \quad (8.1)$$

where $z_{\text{random}}$, $z_{\text{hit}}$ and $\sigma_{\text{hit}}$ are tuneable parameters. $z_{\text{random}}$ is used for a uniform distribution across the map to simulate sensor errors, dynamic objects and errors of the prior map. $z_{\text{hit}}$ and $\sigma_{\text{hit}}$ on the other hand represent a normal distribution around occupied cells in the map. $\sigma_{\text{hit}}$ can be not only be used to simulate the accuracy of the sensors but also to simulate inaccuracies of the prior map.

The probability of a scan is then

$$p(z_t|x_t,m) = \prod_{i=1}^{N} p(z_{ij}^t|x_t,m) = \prod_{i=1}^{N} p(c_{kl}|m) \quad (8.2)$$

with $c_{kl}$ being the cell the detection $z_{ij}^t$ falls into if the pose of the robot is $x_t$. With this formula we can define the weight $w_q$ of the particle $q$ at the time $t$ as

$$w_q = p(z_t|x_t^q,m). \quad (8.3)$$

To avoid unnecessary resampling operations we first calculate the effective sample size (ESS) [28]

$$\text{ESS} = \frac{N}{1 + cv^2} \quad (8.4)$$

where $cv^2$ denotes the variation of the weights of the particles

$$cv^2 = \frac{\text{var}(w)}{\mu(w)} = \frac{1}{N} \sum_{i=1}^{N} (N \cdot w_i - 1)^2 \quad (8.5)$$

with $\text{var}(w)$ being the variance of the weights and $\mu(w)$ being the mean of the weights. If the ESS drops below a percentage of the sample size

$$\text{ESS} < BETA \cdot N \quad (8.6)$$

the particle pool contains too many unlikely samples and resampling is necessary.

During resampling the particle weights are normalized, so that all weights sum up to 1. Then the numerical range between 0 and 1 is mapped towards the particles in a way that each particle occupies a length according to its weight. After that $N$ numbers between 0 and 1 are generated and for each number a copy of the particle, this function maps to, is generated. While it is possible to simply use $N$ random values, a more reliable behaviour can be achieved by generating only a single random value between 0 and $1/N$ and placing the other $N-1$ numbers at $1/N$ intervals after this random number. This strategy is also known as low variance resampling.

In this work we will be the low variance resampling strategy, with logarithmic weights instead of the weights directly, together with the likelihood fields according to Thrun et al. We are using logarithmic weights to decrease the influence of single outliers, which is especially important due to the high noise of the sensors. Low variance resampling is used since it is the most robust strategy to keep a diverse particle pool alive.
Chapter 9
Results

9.1 Test setup

Figure 9.1: Field of view of the agent

To test our implementations we use a Truck with 4 77 GHz radar sensors. Their Field of View is shown in Figure 9.1. The front radars and the stereo camera are not used. Radars in the 77 GHz range are already used in many modern vehicles for other applications like collision avoidance mechanisms, parking support and driving in convoy. While in these areas the possibility to get the speed and direction of the detections make radars a highly interesting technology, for SLAM the variance and low amount of valid detections presents additional problems. However, if these problems can be overcome, the presence of such radars in modern vehicles may enable a widespread use of SLAM on these platforms without additional costs.

As additional problems, the radars are not synchronized and do not provide range data in fixed time intervals. The truck does also have, aside from the RTK GPS, no accurate enough odometry information. To focus on the main goals of this work the RTK GPS is used to calculate a very accurate odometry reading and draw samples from these estimations according to the Gaussian probabilistic motion model as input for the SLAM algorithm. Before drawing a sample the detections of the sensors are merged into 1 second frames with the accurate odometry estimation from the RTK GPS. At such intervals the frames typically have around 100 to 200 detections and the error of the odometry is small enough to be considered negligible.

Then a basic Graph-SLAM is built with the artificial odometry data. The nodes of the graph store the most likely poses of the robot and the edges between the poses
store the odometry values and the covariance matrix representing the likelihood field around the pose. In addition, we use NDT scan registration for loop detection. However, instead of trying to match 2 scans directly first a submap is built from the last 2 seconds using the artificial odometry readings to join the input frames. This submap is then registered to the map built by Graph-SLAM. For the initial guess for the NDT registration all positions in the graph in a 10m radius around the current estimate pose are tried. For the orientation however the current estimate is used. Once again, the error of the odometry accumulated over this time should be low enough to provide a good estimation of the true scan. A fixed radius for the pose error is possible since a prior map will be used to bound the error of the pose of the robot when no loop was detected for some time. Building a submap from the input was mainly implemented to be able to change the duration of the input frames. To avoid matching this scan to itself or other newer scans we build the map for this use with a delay. The resolution of the NDT map is a grid with 1x1 meter cells. To avoid adding constraints if the initial guess was too far off, to produce a correct result, the score of equation 3.9 is first normalized by dividing it through the number of points in the 2 seconds submap. Then we only consider scan registrations with a normalized score of 0.2 or less. If a loop closure is detected this way we add a constraint between the pose of the scan and the pose used as initial transformation. As covariance matrix a fixed covariance matrix with the values

\[
\begin{pmatrix}
0.000001 & 0 & 0 \\
0 & 0.000001 & 0 \\
0 & 0 & 0.00000001
\end{pmatrix}
\]  

(9.1)
is used. The reason for these values is, that constraints through scan registration will be the most accurate constraints in the graph. With such low values a successful loop detection will take priority over all other constraints. It might also be possible to use the Hessian matrix of the NDT registration for a better covariance estimate, but when used directly the covariance is generally too big to cause visible improvements of the map.

To extract absolute information from the prior map we are using MCL with a cached likelihood field and low variance resampling. To take account of the lack of details in the prior map the standard deviation of the normal distribution around obstacles will be overestimated on purpose. This leads to a wider variety in the particles and, therefore, a bigger covariance matrix around the estimated position. Since the runtime of the MCL is negligible in comparison to the scan registration and graph optimization, we are using a fixed sample size \( N \) of 1000. To simplify the localization problem we also assume the initial position on the prior map is already known. It would also be possible to spread the initial particles over the whole map to find the real position through resampling. However, for this to work reliable we would have to use a much bigger sample size, which would then unnecessarily slow down the MCL algorithm once the position was found or we would have to work with a dynamic sample size. The remaining parameters of the MCL are

\[
\begin{align*}
\sigma_{hit} &= 1m \\
z_{hit} &= 0.99 \\
z_{random} &= 0.01 \\
BETA &= 0.7
\end{align*}
\]  

(9.2)

Unfortunately, the MCL algorithm works under the assumption that the amount of detections from the sensor per frame is constant. With our radar sensors, however, the variance of valid detections may lead to an erratic behaviour of the resampling algorithm, since additional points would change the difference between a good match and a bad match by a high factor. However, this problem can be solved by scaling the weights of the particles to a virtual detection count \( N_v \).


\[ w_q' = (w_q)^{N_w} \]  \hspace{1cm} (9.3)

where \( N \) represents the actual amount of detections.

Since the localization worked without this change, the MCL algorithm was left unchanged in this work.

### 9.2 Simulation results

We already included a map of the test data without adding additional noise in Figure 7.2 and a picture of the prior map, which we will use with the MCL algorithm, can be seen in Figure 7.1. The area was explored starting at the bottom entry of the map driving around the building in the middle to the right in anticlockwise direction then going up to the area in the top left and then turning and going back towards the beginning, passing the building in the middle to the right on the left side. The test data is 4 minutes long. To test the scan registration we added noise according to the Gaussian Motion Model with the parameters

\[
\begin{align*}
\alpha_1 &= 0.01 \text{ meters/meter} \\
\alpha_2 &= 0.005 \text{ meters/degree} \\
\alpha_3 &= 0.25 \text{ degrees/meter} \\
\alpha_4 &= 0.025 \text{ degrees/degree} \\
\sigma_{xy}^{\min} &= 0.04 \text{ meters} \\
\sigma_{\phi}^{\min} &= 0.2 \text{ degrees}
\end{align*}
\]  \hspace{1cm} (9.4)

Figure 9.2: Test data for scan registration

A map built with the noisy odometry data can be seen in Figure 9.2. While the top of the map looks quite clean and does not show any obvious errors it is easy to recognize a rotational error in the building in the middle to the right. In this area there are also less obvious features affected by this rotational error around the building, which increases the chance of a successful scan registration and its accuracy.
In Figure 9.3 we can see the result of applying scan registration to the noisy input data. We can see that the building in the middle now shows no signs of being recorded twice. There is still some visible error on the wall at the slope in the bottom right, but the improvement compared to the noisy data is still huge.

However, since no pairwise scan registration is performed, only errors that accumulate inside a loop can be minimized and the resulting map will only be plausible inside such loops. If the robot moves in a pattern that has no loops the error of the map can quickly increase and lead to a state where even scan registration would not work with a local search approach. However, even in our example the corrected map is a lot less accurate than it looks like. This can be seen if we project it onto the building plan of the area we already presented in Figure 7.1. In Figure 9.4 we can see that the result of the scan registration does not fit the building plan. This is caused by an error in the
rotation before the beginning of the loop detected through scan registration. This error
can not be corrected by scan registration.

To allow the use of local scan registration algorithms and improve the overall qual-
ity of the resulting map we decided to extract information from a prior map. Due to
the test data being characterized by an initial seed for the random number generator
and the SLAM algorithm running at the same time as the creation of the test data, the
use of MCL changed the input map, since the sampling of MCL will change the inter-
nal random seed and lead to different sampling for the artificial odometry. However,
since the positive effects of using a prior map on the overall quality can also be seen
on different input data and works stable independent of the random seed, we decided
to also change the motion model parameters and show the effect on another input. The
motion model parameters used for the algorithm with MCL are

\[
\begin{align*}
\alpha_1 &= 0.01 \text{ meters/meter} \\
\alpha_2 &= 0.005 \text{ meters/degree} \\
\alpha_3 &= 0.5 \text{ degrees/meter} \\
\alpha_4 &= 0.05 \text{ degrees/degree} \\
\sigma_{xy}^{\text{min}} &= 0.04 \text{ meters} \\
\sigma_{\phi}^{\text{min}} &= 0.2 \text{ degrees}
\end{align*}
\]  

Since the MCL will bound the error of the pose it is possible to work with a less
accurate odometry.

Figure 9.5: Raw test data with changed error values and seed used for MCL in the following
Figures

The test data created with these inputs can be seen in Figure 9.5. While in this
case the open area in the top left seems to be more accurate due to the rotational errors
negating themselves, the building in the middle to the right is hardly recognizable
anymore. However, if MCL is run on the prior map and these new constraints are
added the map shows a drastic improvement, which can be seen in Figure 9.6.

Even though in a lot of places the contours of the obstacles seem a bit blurred, the
map fits the prior information and all obstacles are clearly recognizable. The only area
where an obvious error can be seen is the wall of the slope in the bottom right again.
This can be explained by the lack of information on the prior map in that area and the
inaccuracy of the building in the middle to the right. These effects can lead to a larger
covariance matrix of the MCL constraints, which means that the errors of the noisy
odometry data are less likely to be corrected.
CHAPTER 9. RESULTS

Figure 9.6: Test data from Figure 9.5 with added constraints from running MCL on the prior map but without loop detection

Figure 9.7: Test data from Figure 9.5 with added constraints from MCL and loop detection

Finally, we also run scan registration together with MCL on the noisy data, since scan registration does not change the random seed. The result can be seen in Figure 9.7. It can be seen that the resulting map shows sharper contours of the objects and still fits the prior map very accurately. The error at the wall in the bottom right was also corrected by the scan registration. However, if we compare it to the result of using scan registration alone we can see that the contours in the top left are less sharp. This is mostly due to the increased noise parameters in the motion model, but also due to the fact that the obstacles of the prior map in this area contain almost no details.

Since the development of the algorithm presented here was on a virtual machine running Linux and the focus was on creating a working algorithm rather than online capability the result is rather slow. On a Intel i5-2500 (3,3 GHz) with Windows 7 and a virtual machine with Ubuntu the final run with MCL and scan registration took 25 minutes to process, while the input was only 4 minutes long.
Chapter 10
Conclusions

In this work we presented a complete Graph-SLAM algorithm, which extracts pose estimates from a rough map and adds them as additional constraints. In comparison to other papers we only used radar sensors which are legal to use in public street traffic.

We showed that Graph-SLAM works well with radar sensors, even though there is no fixed amount of detections, and that prior information in form of absolute poses from a localization algorithm can significantly improve the result of Graph-SLAM and easily be combined with scan registration. We also showed that NDT scan registration, even though it was developed as a way to handle dense point clouds, works well on sparse data as well in the context of Graph-SLAM.

While we would have liked to present results with real odometry values we are confident that the simulation results show that the algorithm could be used on a robot with odometry readings instead of an GPS sensor.

However, in the field of robotics it is important to weight the use of the algorithm off against the negative impact it may have on society. While our algorithm does not actually perform any actions and only gathers information, this information is mainly of interest so that the robot can later on perform informed actions.

Since GPS sensors are already present in most areas on earth and provide a simpler solution to the SLAM problem our algorithm is most likely to be used in space programs, indoor areas, underground areas and other areas that have not enough satellite coverage to calculate a proper GPS position. We can assume that an application in underground areas or in a space program will prevent humans from having to work in these, in most cases, hazardous areas. These are, obviously, desired applications. While the use in indoor environments and areas without enough satellite coverage may as well be of use in the case of catastrophes, here our algorithm may lead to robots replacing humans in perfectly safe working places. However, we can assume that, with some additional effort, a map may be built in these safe areas anyways and as such our work does not have a strong influence on this topic. We do not expect our algorithm to be of use for any military operations, since the military will in most cases have sufficient satellite coverage in outdoor areas and in indoor and underground areas real time information will be much more valuable since an undisturbed exploration run through a building is very unlikely. We also assume that military operations on extra terrestrial bodies will not take place any time soon and leave the discussion about it to another generation.

For future works we propose to validate the results of the simulations of this work with real odometry data and to optimize the code to run in real time even on larger maps. For example the 10m radius for the scan registration is quite time consuming and it is probably not necessary to add MCL constraints for every pose since the error that will accumulate between two poses is rather low. In addition, closed loops in areas the agent is not currently exploring could be temporarily removed from the
CHAPTER 10. CONCLUSIONS

graph to only optimize the entry and exit points of the loop with EKF estimations for the constraints between these nodes.
Bibliography


