Master Thesis

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Localization of Mobile Robot Using Multiple Sensors

David Nováček

Supervisor: Ing. Vladimír Smutný, Ph.D.
Field of study: Cybernetics and robotics
May 2019

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university thesis.

Prague, 24. May 2019

Abstract

This thesis is dedicated to a lifelong localization of a mobile robot, which is equipped with the multiple sensors. The information about the robot position and the map are necessary for the autonomous movement.

The goal of this thesis is implementing the method based on the Normal Distribution Transform for solving the problem called Simultaneous localization and mapping. The important requirement is the ability to use the CAD drawing of the environment as an initial map.

The thesis contains the principle of the method, the description of the implementation, and the experiments evaluation. The experiments have been focused on the difference between the localization and mapping process with and without the CAD drawing.

Keywords: localization, mapping, robot, lidar, SLAM, normal distribution transform, NDT, scan matching, CAD drawing, ROS

Supervisor:
Ing. Vladimír Smutný, Ph.D.

Abstrakt

Tato práce se věnuje celoživotnímu určování polohy mobilního robotu, který je vybavený různými senzory. Informace o poloze robotu a mapa jsou nezbytné pro zajištění autonomního pohybu.

Cílem je implementovat metodu řešící problém zvaný Simultání lokalizace a mapování pomocí přístupu využívající Transformaci normálního rozdělení. Důraz je kladen na schopnost využít CAD výkresy prostředí jako počáteční mapu.

Práce zahrnuje princip metody, popis implementace a zhodnocení výsledků, které bylo zaměřeno na rozdíly v lokalizaci a mapování s využitím CAD výkresů a bez nich.

Klíčová slova: lokalizace, mapování, robot, lidar, SLAM, transformace normálního rozdělení, NDT, scan registrace, CAD výkres, ROS

Překlad názvu: Určování polohy mobilního robotu na základě informací z různých senzorů

Contents

Figures

Tables

Chapter 1

Introduction

One of the most fundamental problems in mobile robotics is localization. Computation of the robot position requires a model (a map) of the environment. At the same time, building the map needs information about the position of the robot. This chicken-egg problem is called Simultaneous Localization And Mapping, and it is commonly abbreviated as SLAM. In contrast with the localization with a known map or the mapping with a known position, in SLAM problem, the map and position are both unknown.

Localization and mapping are essential for other applications such as navigation, path planning, or multi-robot coordination. For these reasons, the software, which is able to work with live data and provides the location of the robot, is required.

The sensors provide the pieces of evidence for solving SLAM problem. The range finders sensors (e.g., radar, lidar) are commonly used as well as the cameras. Information from these sensors has to be processed by the algorithm solving SLAM.

The algorithms are commonly focused on one-off experiments evaluation. The scenario has a character of the expedition, and long term usage is neglected. This work is focusing on the implementation of the algorithm, which is able to operate life-long. Our motivation is to provide the localization for a ground vehicle, which operates in the long term on the shop floor.

In the last couple of years, the method based on Normal distribution transform has proven to be a suitable solution for scan registration, localization and mapping. The NDT method can handle the dynamic object in the environment and is also able to processed noisy data from range sensors. Last but not least advantage is the fact that measured data from range sensors do not need to be stored. This fact allows life-long use of the algorithm.

This work is divided into six chapters. The second chapter defines SLAM problem. The used methods are presented in the third chapter. The fourth chapter provides implementation information. The fifth chapter describes experiments and results, and the last chapter involves summarization and suggests improvements.

Chapter 2

Problem analysis

This chapter defines the problem called Simultaneous Localization And Mapping. It will describe models for the map representation, common issue with the solution and our scenario will be presented.

2.1 SLAM problem definition

As it is mentioned in the previous chapter, the SLAM problem is more challenging than pure localization or mapping, since the map and position of the robot are both unknown. The error in the pose estimation correlates with the error of the map. From a probabilistic perspective, there are two main forms of the SLAM problem [1]. On-line SLAM estimates the probability of position in the actual time t

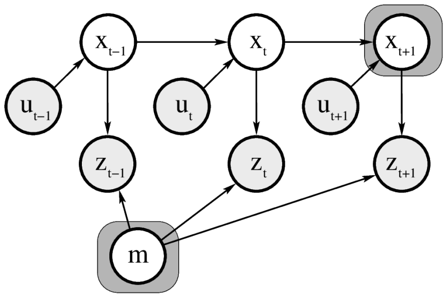
$$p(x_t, m|z_{1:t}, u_{1:t}), \tag{2.1}$$

where x_t is an actual position of the robot, m is a map, $z_{1:t}$ is a sequence of the measurements and $u_{1:t}$ is a sequence of the controls. The estimation of only actual position allows discarding the previous measurement.

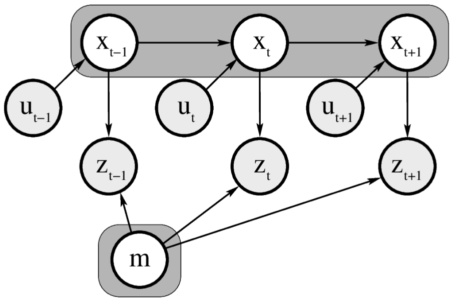
The other type of SLAM is called Full SLAM. It is a problem where all positions $x_{1:t}$ are estimates (a trajectory reconstruction)

$$p(x_{1:t}, m|z_{1:t}, u_{1:t}). \tag{2.2}$$

The Full SLAM problem is a more challenging task for computing in real time. On the other hand, the full SLAM can recover from failures, because there are all measurements available.



(a) : On-line SLAM



(b) : Full SLAM

Figure 2.1: SLAM described by the dynamic Bayes network. The gray squares mean an unknown states (the position of the robot, the map). The picture is from [1]

2.2 Localization part

The localization has two main scenarios. In the first case, the initial position of the robot is known, and thus, it is possible to follow the known position. This problem is called continual localization [5]. The continual localization is solved by correction of the odometry error by scan matching. The odometry sensor suffers from significant error over time. In case the scan matching cannot correct the error, then the divergence occurs, and the correct estimation of the robot position is lost. Generally, the robot is not able to recover from this state.

The second case is a global (absolute) localization [5]. In this scenario, a priori information about the initial robot position is not known. This situation occurs, when the sensor system of the robot fails, or when the robot starts and no information about the initial position is provided. The solution is based on searching for matches among the measured data and the map. The absolute localization is a more difficult problem than the continual localization because the large state space has to be examined. It is also more prone to failure e.g., the robot is inside one of the many similar rooms.

The number of parameters required to determine the robot position depends on the number of degrees of freedom of the robot. In case of the 2D localization, it is Cartesian coordinates (t_x, t_y) of the robot to the reference point and the robot heading angle θ. These three parameters $\vec{p} = [t_x, t_y, \theta]^T$ describe the robot position completely.

2.3 Mapping part

The other part of the SLAM problem is the building of the map. The model of the environment where the robot operates is necessary for path planning and collision avoidance. The mapping process has to deal with the inaccurate sensors and the noisy data while the map has to be consistent, actual and accurate. The inaccuracies in the map can lead to poor localization. There are also requirements for time and memory consumption of the algorithm. The robot should provide the map and information about self-position in real time. The compromise between accuracy and time consumption cannot be avoided.

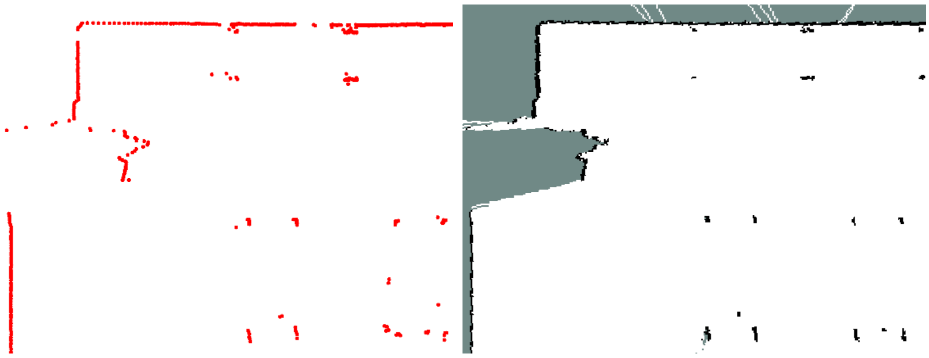
There are many ways to represent the map. One of the simples models is a Point cloud map [6]. The output data from the range sensors can be expressed as a point cloud. The map creation is done by simply storing these measurements in the one large point cloud. This model is suitable for an Iterative closest points (ICP) algorithm [7]. The disadvantage is memory consumption because the amount of data increase over time.

The golden standard model is an Occupancy grid [8]. The Occupancy grid discretizes the environment into a regular square grid of cells. Each cell stores a value of probability that the cell is occupied. The advantage is constant memory consumption over time. The required memory is given by the mapping area size. Another advantage is the ability to represent free space explicitly which is important for path planning. Occupancy grid is also used as the standard for representing map in the Robot Operating System (ROS).

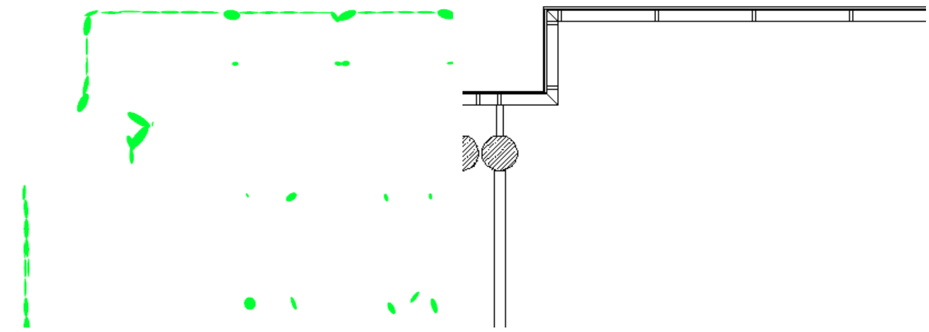
The main model used in this work is the NDT map. Similarly, as Occupancy grid, the NDT map discretizes the environment into the grid of cells. Each cell approximate measured data with a Gaussian [9]. The points inside cell are used for computing mean vector and covariance matrix $(\vec{\mu}, \mathbf{\Sigma})$. The NDT representation is introduced in chapter 3 with more detail.

It might happen that prior information about the environment exists. This information can be described as a CAD drawing, and it can be used as an initial map for the robot before the mapping process starts. The transformation from the CAD drawing into the NDT representation has been done by Pánek [4]. The converted map in the NDT representation can be used as the initial map for the localization and mapping process.

In the mapping task, there are known issues as the repetitive environment, long corridor, and loop closure detection. The repetitive environment is challenging for the localization because the current scan in scan matching can be registered to more than one position. The long corridor is an example of the repetitive environment. In this situation, the actual measured data corresponds to many positions. The uncertainty rise in the longitudinal direction of the corridor. In this situation localization is failing very often. The inconsistencies in the map are created and the whole process of localization and mapping is collapsing. The other problem is the loop closure. The loop closure reflected the situation when robot visits already mapped area, but with different direction (the crossover).



(a) : Point cloud map **(b) :** Occupancy grid map



(c) : NDT map **(d) :** CAD drawing

Figure 2.2: Different models of a map describing the same environment. The Point cloud map and the NDT map are not able to represent free space explicitly. The Occupancy map is the standard way to describe the environment. The CAD drawing is unusual in the mobile robotics, and this model is processed to other representation such as NDT map.

2.4 Scan registration

The scan registration is a method primarily used for finding the transformation between two scans. However, the scan registration can be used for the reconstructing robot position and therefore it is one of the crucial parts of the solution for the SLAM problem. Maximization of the likelihood of i-th pose and a map are computed relative to the (i-1)-th position and a map [10].

$$\vec{\hat{x}}_t = \arg\max_{\vec{x}_t} \{p(z_t|\vec{x}_t, \hat{m}^{[t-1]})p(\vec{x}_t|u_{t-1}, \vec{\hat{x}}_{t-1})\}, \tag{2.3}$$

where $\vec{\hat{x}}_t$ is an optimal actual pose and $\hat{m}^{[t-1]}$ is an optimal map in the time $(t-1)$. In the next step, an optimal map $\hat{m}^{[t]}$ is estimated and the whole process is computed again with other pair of the scans.

One of the most common algorithms for scan registration is an Incremental scan matching. It can be done with the Iterative closest point (ICP) algorithm. The ICP algorithm is looking for the transformation between two scans by minimizing distances between pairs of corresponding points (each from the one scan). A similar principle is used in the NDT approach, which is specified in chapter 3. Both of these algorithms iteratively look for the optimal transformation. An optimization method like Newton or Levenberg-Marquardt is used. These methods require an initial position estimation. If the initial estimation is not in the convergence area, the optimization can be stuck in the local minimum which leads to the incorrect scan registration and failure of the localization.

2.5 Our scenario

The robot (ground vehicle) is autonomously moving with the load (boxes, pallets) on the factory shop floor. The robot is equipped with a different type of sensors such as lidar, camera, and odometry. This thesis works only with the lidar and odometry data.

The goal is to implement an algorithm which solves on-line life-long 2D SLAM using the NDT approach. Since the environment where the robot operates is known, the CAD drawing can be used as the initial map. The map correspond with horizontal cut of the building in the height of the lidar position.

The algorithm implementation should

- be able to import map based on the CAD drawing of the building,
- compute localization of the robot and the map in real time,
- follow standard interfaces in the Robot Operating System (ROS).

Chapter 3

Used methods

This chapter describes methods and algorithms based on the Normal distribution transform (NDT) used in this thesis. The NDT scan registration and the map building will be introduced.

3.1 Normal distribution transform of Point cloud

The natural way of keeping output data from range sensors (sonar, lidar) is the point cloud. However, as long as the robot senses new data, the point cloud can be expensive to store. This thesis aims for the long-term localization, so a memory needs to be bounded. Bieber and Straßer [9] have introduced an alternative representation of the map. The map can be 2D or 3D without changing the method [11]. The map consists of the regular grid of cells. Each cell contains a single Gaussian representing surface of the measured object, hence the Normal Distribution in the approach name.

The measured points belonging to the cell are thus represented by their distribution approximated by the normal distribution. When some additional information is stored (e.g., the number of measured points in the cell), the Gaussian can be updated without the need for accessing the previous measurements.

The transformation of the input point cloud to the NDT grid follows two steps:

1. The input Point cloud (2D scan from lidar) is subdivided into the regular grid. Points from the scan are separated into the corresponding cells.
2. The mean vector $\vec{\mu}$ and the covariance matrix $\mathbf{\Sigma}$ are computed for each cell as

$$\vec{\mu} = \frac{1}{m} \sum_{k=1}^{m} \vec{y}_k, \tag{3.1}$$

$$\mathbf{\Sigma} = \frac{1}{m-1} \sum_{k=1}^{m} (\vec{y}_k - \vec{\mu})(\vec{y}_k - \vec{\mu})^T, \tag{3.2}$$

   where $\vec{y}_k = [x, y]^T \in \mathbb{R};\ k \in \{1, \ldots, m\}$ are a coordinates of the points and m is the number of points in the cell.

A set of local probability density functions, one per cell, is obtained. The approach using normal distribution transform brings following advantages.

- The required memory is constant over time and it is given by the mapped area and the cell size.
- The Gaussian represents smooth surface in contrast with the point cloud.
- The probability density function of the normal distribution is continuous and smooth and has a continuous and smooth first and second derivative, which can be computed analytically. This attribute is useful in scan matching, where gradient needs to be found for the Newton's optimization [2].

The measured Point cloud and the resulting NDT grid are visualized in Fig. 3.1. The Gaussian is not created when the cell contains less than five points. The cell can hold only one Gaussian. The Gaussian is nonzero up to infinity, but the cells are considered independent in the NDT approach.

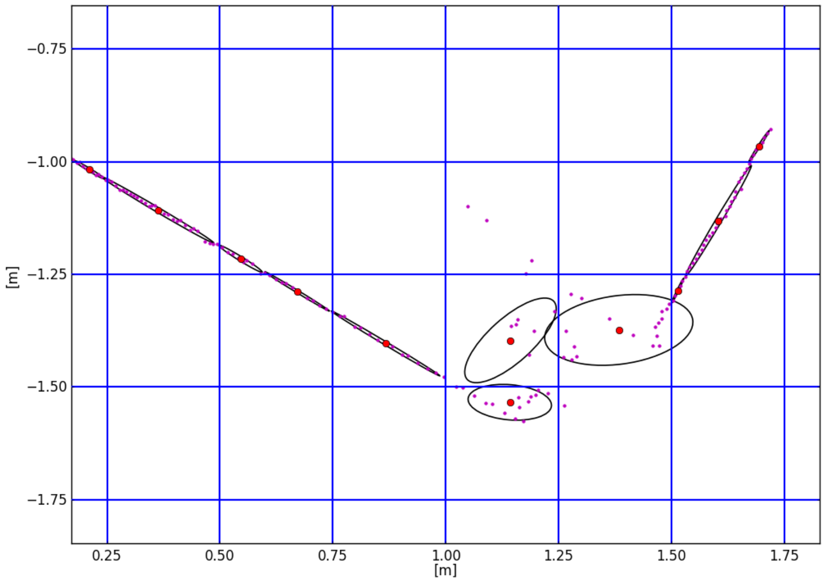


Figure 3.1: The NDT grid computed from the laser scan data. The blue lines borders cells. The magenta dots are the measured points from the laser scan. The black ellipses correspond with 80% mass of the Gaussian. The red dots are the mean values of cells. The cell size is set to 0.25 m.

3.2 NDT scan matching

The goal of scan matching is to find the transformation between two scans. Since the NDT representation of the point cloud is available, there are three scenarios of the scan matching. A Point cloud to Point cloud (**PCL-to-PCL**) registration is a standard way of the scan matching. The Iterative Closest Points (ICP) algorithm [7] minimizes the distances between pairs of points. This is a well-known solution to point-to-point registration. A Point cloud to the set of distributions (**PCL-to-NDT**) registration is a technique using the NDT grid instead of the reference point cloud. This chapter is dedicated to PCL-to-NDT registration and it will be described in more details in this section. A set of distributions to the set of distributions (**NDT-to-NDT**) registration is the last possibility of the scan matching. In this case, the registration problem is interpreted as minimization of the distances between two NDT grids [11].

Bieber and Straßer have introduced the algorithm for the PCL-to-NDT registration in [9] and later they improved it in the paper [12]. Magnusson [2] provides a comprehensive description of the NDT approach. The PCL-to-NDT registration works with a point-cell pair. It is possible to work with the source scan (the newer one) and the target scan (the older one) or with the source scan and the NDT map. It is the same problem in the principle, the NDT representation is essential. The robot position is described by three parameters $\vec{p} = [t_x, t_y, \theta]^T$ as was mentioned in section 2.2. The 2D transformation function is defined as

$$T(\vec{p}) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \vec{x} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}. \tag{3.3}$$

The equation Eq 3.3 [2] describes transformation of the point $\vec{x} = [x, y]^T$ from the source scan to the map (or to the target scan).

In the original work [9], the points from source scan were simply subdivided into the NDT map cells with the current transformation. This simple approach works only if the source scan and the NDT map has misalignment similar to the cell size. For this reason, the nearest neighborhood cell is found as the corresponding cell for the point-cell pair.

The algorithm of the PCL-to-NDT registration follows these steps [9]:

1. Initialize parameters of the transformation $\vec{p} = [t_x, t_y, \theta]^T$ by zero or using the odometry position estimation.
2. Transform every point in the source scan using transformation (Eq 3.3) with the current $T_i(\vec{p})$. The T_i means the transformation in the i-th iteration of the algorithm.
3. Create the point-cell pairs from the transformed points and the NDT map.
4. Compute a value of a score function Eq 3.10 described in a section 3.2.1.
5. Optimize the parameters $\vec{p}$ by Newton's method described in a section 3.2.2 and find the new transformation T_{i+1}.
6. Go to step 2 until a convergence criterium or a maximum number of iterations is reached. The convergence criterium is reached when the absolute value of the difference between the value of the score function Eq 3.10 in the time t_i and t_{i+1} is smaller than the threshold.

3.2.1 Score function

Bieber and Straßer [9] used the probability density function of the normal distribution as a score function for one point. Assuming that the point creating the NDT cell was generated by a two-dimensional normal random process, the likelihood of having measured $\vec{x}$ is

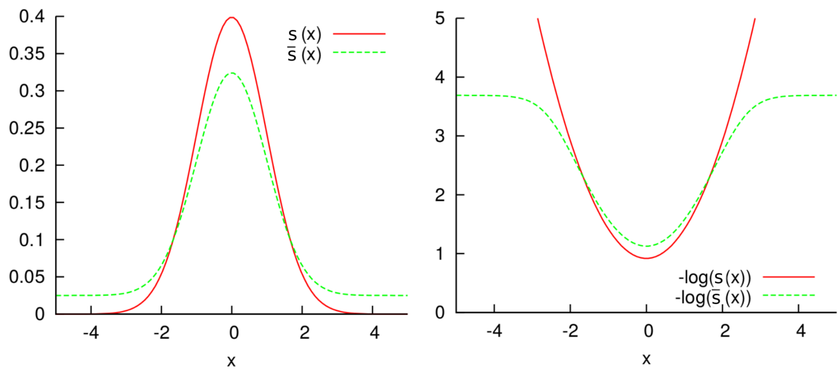
$$s(\vec{x}) = \frac{1}{\sqrt{(2\pi)^2|\mathbf{\Sigma}|}} \exp\left(\frac{-(\vec{x}-\vec{\mu})^T\mathbf{\Sigma}^{-1}(\vec{x}-\vec{\mu})}{2}\right), \tag{3.4}$$

where $(\vec{\mu}, \mathbf{\Sigma})$ are the mean vector and the covariance matrix of the cell. The factor $(\sqrt{(2\pi)^2|\mathbf{\Sigma}|})^{-1}$ is the normalization constant, where $|\mathbf{\Sigma}|$ is the determinant of the Covariance matrix $\mathbf{\Sigma}$.

The probability density function can be modified for the outliers rejection. The negative log-likelihood of a normal distribution grows without bound for points far from the mean. It can be seen in Fig 3.2b. The outliers in the scan have a huge influence and negative impact on the result [2]. The score function, which approximates the probability density function and inhibits the influence of the outliers is described as following

$$\bar{s}(\vec{x}) = c_1 \exp\left(\frac{-(\vec{x}-\vec{\mu})^T\mathbf{\Sigma}^{-1}(\vec{x}-\vec{\mu})}{2}\right) + c_2 r_0, \tag{3.5}$$

where r_0 is the expected ratio of the outliers and c_1 and c_2 are the scaling constants.



(a) : Likelihood **(b) :** Negative log-likelihood

Figure 3.2: Comparing a normal distribution $s(x)$ (red) with the mixture model $\bar{s}(x)$ (green). The negative log-likelihood function $-log(\bar{s}(x))$ is used as score function for one measured point. Figured is taken from [2].

The function $\bar{s}(\vec{x})$ Eq. 3.5 can be approximated using probability density function by setting parameters

$$d_3 = -\log(c_2), \tag{3.6}$$

$$d_1 = -\log(c_1 + c_2) - d_3, \tag{3.7}$$

$$d_2 = -2\log\left(\frac{-\log(c_1\exp(-\frac{1}{2}) + c_2) - d_3}{d_1}\right), \tag{3.8}$$

so that new function

$$\tilde{s}(\vec{x}_k) = -d_1 \exp\left(\frac{-d_2(\vec{x}_k - \vec{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\vec{x}_k - \vec{\mu}_k)}{2}\right), \tag{3.9}$$

where $\vec{\mu}_k, \boldsymbol{\Sigma}_k$ are parameters of the appropriate NDT cell for the point $\vec{x}_k$, has same value as the function $s(\vec{x})$ *Eq.* 3.4 in a points $x = 0, x = \sigma$ and $x \to \infty$ [11]. The score function $\tilde{s}(\vec{x})$ is more suitable for a simpler derivatives than function $\bar{s}(\vec{x})$ *Eq.* 3.5. The function $\tilde{s}(\vec{x})$ still holds the required properties when it comes to the optimization.

The score function with a set of points $X = \{x_1, x_2, \ldots, x_n\}$ (i.e., the point cloud) and the transformation $T(\vec{p})$, which transform point $\vec{x}$ by Eq. 3.3, is defined as a sum of values of the score function $\tilde{p}(\vec{x})$ Eq. 3.9.

$$s(\vec{p}) = -\sum_{k=1}^{n} \tilde{s}(T(\vec{p}), \vec{x}_k). \tag{3.10}$$

3.2.2 Newton's algorithm

The optimal parameters $\vec{p}$ can be found by minimizing value of the score function Eq. 3.10. The minimization is done by iterative solving of the equation

$$\boldsymbol{H}\Delta\vec{p} = -\vec{g}, \tag{3.11}$$

where $\boldsymbol{H}$ is a Hessian matrix of a partial derivatives $H_{i,j} = \frac{\partial^2 s}{\partial p_i \partial p_j}$ and $\vec{g}$ is a gradient of the function $s(\vec{p})$ Eq. 3.10 [2]. An every iteration looks for new $\Delta\vec{p}$ which is added to the current parameters ($\vec{p} \leftarrow \vec{p} + \Delta\,\vec{p}$). The gradient computation $\vec{g}$ describes following equation

$$g_i = \frac{\partial s}{\partial p_i} = \sum_{k=1}^{n} d_1 d_2 \vec{x}'^T_k \boldsymbol{\Sigma}_k^{-1} \frac{\partial \vec{x}'_k}{\partial p_i} \exp\left(\frac{-d_2}{2} \vec{x}'_k{}^T \boldsymbol{\Sigma}_k^{-1} \vec{x}'_k\right), \tag{3.12}$$

where $\vec{x}'_k$ is the transformed point $\vec{x}_k$ with the current parameters of the transformation $\vec{p}$, and the point is shifted by $\vec{\mu}_k$ for placing relative to the center of the appropriate NDT cell

$$\vec{x}'_k \equiv T(\vec{p}, \vec{x}_k) - \vec{\mu}_k. \tag{3.13}$$

And lastly, the Hessian matrix $\boldsymbol{H}$ is computed

$$H_{i,j} = \frac{\partial^2 s}{\partial p_i \partial p_j} =$$
$$\sum_{k=1}^{n} d_1 d_2 \exp\left(\frac{-d_2}{2}\vec{x}'_k \boldsymbol{\Sigma}_k^{-1} \vec{x}'_k\right)\Bigg(- d_2(\vec{x}'^T_k \boldsymbol{\Sigma}_k^{-1} \frac{\partial \vec{x}'_k}{\partial p_i})(\vec{x}'^T_k \boldsymbol{\Sigma}_k^{-1} \frac{\partial \vec{x}'_k}{\partial p_j}) + \tag{3.14}$$
$$+ \vec{x}'^T_k \boldsymbol{\Sigma}_k^{-1} \frac{\partial^2 \vec{x}'_k}{\partial p_i \partial p_j} + \frac{\partial^2 \vec{x}'^T_k}{\partial p_j} \boldsymbol{\Sigma}_k^{-1} \frac{\partial \vec{x}'_k}{\partial p_i}\Bigg).$$

The first derivate $\frac{\partial \vec{x}'}{\partial \vec{p}}$ in the gradient Eq. 3.12 for the 2D transformation $T(\vec{p}, \vec{x_k})$ Eq. 3.3 is a Jacobian $\boldsymbol{J}_2$

$$\frac{\partial \vec{x}'}{\partial \vec{p}} = \boldsymbol{J}_2 = \begin{bmatrix} 1 & 0 & -x_1 \sin\theta - x_2 \cos\theta \\ 0 & 1 & x_1 \cos\theta - x_2 \sin\theta \end{bmatrix}. \tag{3.15}$$

The derivative $\frac{\partial^2 \vec{x}'}{\partial p_i \partial p_j}$ in the Hessian matrix $\boldsymbol{H}$ Eq. 3.14 is calculated as follow

$$\frac{\partial^2 \vec{x}'}{\partial p_i \partial p_j} = \begin{cases} \begin{bmatrix} -x_1 \cos\theta + x_2 \cos\theta \\ -x_1 \sin\theta - x_2 \sin\theta \end{bmatrix}, \text{ if } i = j = 3 \\ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \text{ otherwise.} \end{cases} \tag{3.16}$$

3.3 Mapping step

The result of the scan registration determines the location of the robot (the localization step). The known robot position allows joining the measured point cloud with the NDT map (the mapping step). In the original paper [9], storing of all measurement has been necessary. Recomputing the parameters of the NDT cell (mean vector and covariance matrix) have required the points from the all previously measured scans.

The original method was improved in the papers [13] and [11]. The Recursive Covariance Update (RCU) has been proposed as a solution for merging two NDT cells. The measurements do not need to be stored anymore. This improvement makes the NDT approach more suitable for long-term localization.

There are two sets of points $\{\vec{x}_i\}_{i=1}^m$ and $\{\vec{y}_i\}_{i=1}^n$, where m and n are the number of points of these sets. The first equation describes combined mean vector $\vec{\mu}_{x \oplus y}$ as

$$\vec{\mu}_{x\oplus y} = \frac{1}{m+n}\left(\sum_{i=1}^{m}\vec{x}_i + \sum_{i=1}^{n}\vec{y}_i\right) = \frac{1}{m+n}(m\vec{\mu}_x + n\vec{\mu}_y). \tag{3.17}$$

The second equation describes combined covariance matrix $\mathbf{\Sigma}_{x\oplus y}$ as

$$\begin{aligned}
\mathbf{\Sigma}_x &= \sum_{i=1}^{m}(\vec{x}_i - \vec{\mu}_x)(\vec{x}_i - \vec{\mu}_x)^T, \\
\mathbf{\Sigma}_y &= \sum_{i=1}^{n}(\vec{y}_i - \vec{\mu}_y)(\vec{y}_i - \vec{\mu}_y)^T, \\
\mathbf{\Sigma}_{x\oplus y} &= \mathbf{\Sigma}_x + \mathbf{\Sigma}_y + \frac{mn}{m+n}(\vec{\mu}_x - \vec{\mu}_y)(\vec{\mu}_x - \vec{\mu}_y)^T.
\end{aligned} \tag{3.18}$$

Besides the advantages of throwing away the previous measurement, the RCU can create the grid with a lower resolution. It can be used for the faster path planning computation [11]. The number of points in the cell increase over time and it can cause numerical instability. Moreover, the high number of points in the cell leads to the problem with the dynamic object mapping [11]. The maximum number of points in cell N is bounded by the parameter M

$$N = \begin{cases} n+m, n+m < M \\ M, n+m \geq M \end{cases}. \tag{3.19}$$

The parameter M influences a map adaptation. The small value of M means that the map is adapted for changes very fast. A high value of M means slow adaptation - the previous measurements have higher influence.

3.3.1 Occupancy extension

The Normal Distribution Transform Occupancy Map (NDT-OM) expands the NDT representation about the occupancy value, which can explicitly describe a free space [11]. This extension is motivated by the mapping of the dynamic objects. The dynamic object can disappear, but the Gaussian in the cell persists. Another motivation is free space modeling, which is crucial for path planning. Since this extension has been implemented and has been taken from [3] as it is presented in [11], a description of the NDT-OM is just an overview of the state-of-the-art of this method.

The cell c_i in the NDT-OM is represented by the parameters $c_i = \{\vec{\mu}_i, \mathbf{\Sigma}_i, N_i, p_i(m|z_{1:t})\}$, where $p_i(m|z_{1:t})$ is the probability that the cell is occupied. The eight numbers are needed for the description of 2D NDT cell (two for mean vector, four for covariance matrix, one for the number of points and one for the probability of occupancy). The value of probability is computed by a ray tracing. The ray tracing starts in a lidar position and ends in a point where ray has been reflected from an obstacle. The line intersects cells between the start and end point. Ray modifies the Gaussian shape and the occupancy probability value. The cells along the ray are updated with the low occupancy probability. The last cell is updated with the high occupancy probability. The mathematical description can be found in [11].

Chapter 4

Implementation

This chapter describes the implementation of on-line 2D NDT SLAM. Used libraries will be presented and the implementation will be introduced. The source code of the implementation can be found in an enclosed CD. The code is based on Jelínek's work [3], which uses NDT for a graph based SLAM. Some basic structures and classes have been taken from that work. Our concept of the solution is quite different since it does not work with the graph SLAM and it is required the CAD drawing import.

4.1 Used software

4.1.1 Robot Operating System (ROS)

The Robot Operating System is an open source framework for Linux distribution. It is a software, which provides useful tools for developing software. The ROS associates with many other libraries such as PCL library, OpenCV, MoveIT. The ROS works with node architecture. The node is a program, which can communicate with other nodes using a topics. One of the advantages of the ROS is a standard format of the messages in the topics. For this reason, a different type of software from different developers can be connected without difficulties. The implementation solving SLAM has been developed in ROS Kinetic [14]. The important libraries connected with the ROS are Eigen and PCL library.

Eigen

Eigen library [15] implements operations with a matrices (e.g., multiplication, inverse matrix computation, singular value decomposition).

PCL

The point cloud structure is used as main representation of the measured data the from lidar. The PCL library solves operation with cloud such as transformation to the different coordinate system.

CAD drawing import

The tool for converting the CAD drawing into the NDT representation has been developed by Pánek in [4]. The simple .txt file with the NDT map contains parameters for each NDT cell (mean vector, covariance matrix, occupancy value, and the number of point in the cell).

4.2 Program overview

The input data are the scans from lidar and the odometry data. The ROS package *message_filters* synchronizes messages from topics with the lidar and odometry data. It is important to have a pair of messages from one time. Frequency of the lidar scans could be different from the odometry frequency.

The input point cloud structure is created from the lidar data by *laser_geometry* ROS package. The point cloud is transformed from a coordinate system of the lidar to the base of a robot coordinate system. The point cloud and the odometry are inputs for the *AlgorithmNdtSlam* class, where the NDT scan matching is computed as described in section 3.2.

The scan matching provides the transformation between the current point cloud and the map (the Localization step). This transformation defines the position of the robot, and thus, the point cloud can be registered into the map (the Mapping step). The ray tracing and the RCU are computed for the update of the map as described section 3.3.

The map and the position of the robot are the outputs of the program. The ROS standard messages are used. The map is published to the topic with the *nav_msgs::OccupancyGrid* message and the position of the robot is published to the topic with the *geometry_msgs::Pose* message. The sections below describe the classes of the program.

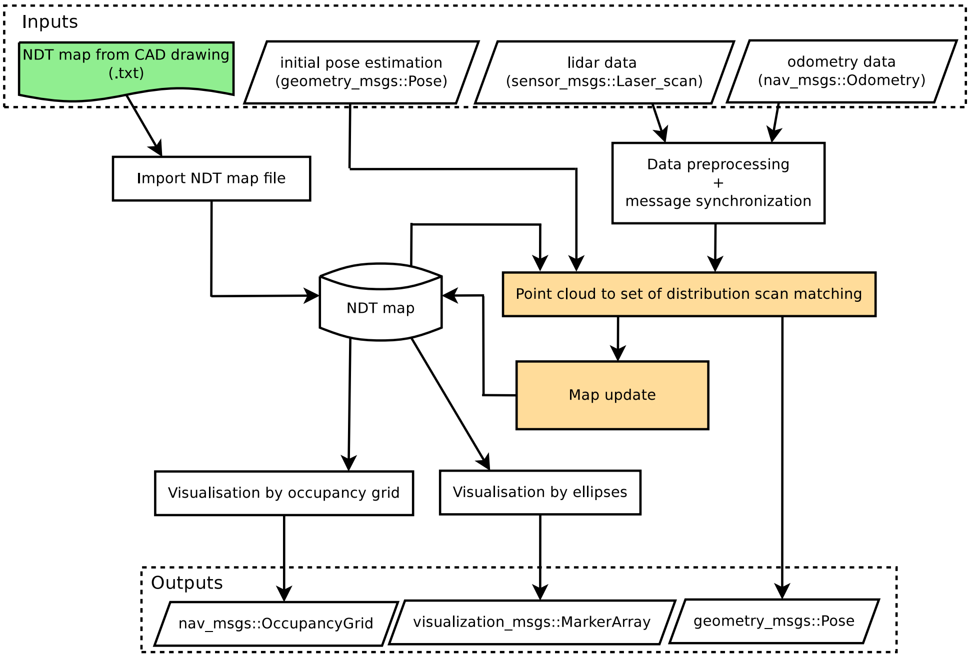


Figure 4.1: Diagram of the program for solving 2D NDT SLAM. The scan registration and the map updating (orange) have been taken from Jelínek [3]. These components have been reworked for our concept of solution. The NDT map file (green) has been prepared by Pánek [4]. The white boxes have been implemented by the author of this work.

4.2.1 NdtSlamNode

The *NdtSlamNode* class encapsulates the algorithm for NDT SLAM to the ROS environment. This class handles the preprocessing of the input data, the time synchronization of the messages from the scan and odometry, and maintenance of the transformations between the map coordinate system and the base of the robot. It is crucial for visualization (RVIZ tool in the ROS). The class creates the ROS node. The *main* function is placed here and the program can be executed in the ROS environments by command *rosrun*. The launch file can be written for the comfortable using. Every pair of the preprocessed data are sent to the *AlgorithmNdtSlam* class, where the data are processed.

The parameters of the class *NdtSlamNode*:

- **robot_base_frame_** - The coordinate system of the base of the robot. The default value is set to *base_link*. (string)
- **odom_frame_** - The coordinate system of the odometry. The default value is set to *odom*. (string)
- **fixed_frame_** - The coordinate system of the map. The robot position is given by the transformation between the *robot_base_frame_* and the

*fixed_frame_*. The default value is set to *map*. (string)

- **laser_topic_** - The name of the ROS topic with the input data from the lidar. The input data format follows the standard ROS message *sensor_msgs::Laser_scan*. The default name of the laser topic is set to *scan_filtered*. (string)
- **odom_topic_** - The name of the ROS topic with the input data from the odometry. The input data format follows the standard ROS message *nav_msgs::Odometry*. The default value name of the odometry topic is set to *odometry/filtered*. (string)

4.2.2 AlgorithmNdtSlam

The *AlgorithmNdtSlam* is the class for the computing 2D NDT SLAM. The memory management for the map and position is done in here. The visualization is also computed in this class. The class publishes to the topics with the output data (the robot position and the map). If the CAD drawing is available, the class deals with the importing of file with the map and sets the initial guess of the robot position.

The class has two important methods called *init* and *update*. The *init* method is called only once when the program is executed. The map is initialized by the CAD drawing or by the first measured point cloud. The coordinate system of the map and the initial position is set.

The *update* method is called with the every pair of the point cloud and the odometry data, which have been synchronized in the *NdtSlamNode* class. The scan matching is computed only if the robot travels required distance or rotates by required angle. The scan matching is not computed when the robot is standing on the one place. The reason is that the measured object by lidar might have the disproportionate number of points in the cells. It might cause a problem in a remapping when the object disappears.

If the robot is moving, the scan matching is computed as described in section 3.2. The result of the scan matching is the transformation between the base of the robot and the map, and this transformation defines the robot position. The map is updated by the merging measured point cloud into the NDT map. It is done by the RCU and the ray tracing as described in section 3.3.

Two types of visualization are available. The first one uses ellipses, which correspond with the Gaussians. The visualization by the ellipses is described in a section 4.2.3 in more detail. The second type of visualization is done by the Occupancy grid and follows the standard ROS message *nav_msgs::OccupancyGrid*. The occupancy map creation is described in section 4.2.4.

The types of visualization can be used independently. The computation of visualization is a time-consuming process. The map is not visualized for each map update, but for a constant time interval. The frequency of the visualization is given by a parameter. The robot pose computed by SLAM is published to the ROS topic *ndt_slam_pose* with the standard ROS message *geometry_msgs::Pose*.

The parameters of the *AlgorithmNdtSlam* class:

- **cell_size_** - The size of the cell in the NDT map in meters. The default value is set to *0.25* m. (double)
- **map_file** - The path to the .txt file with the NDT map for import. (string)
- **import_map_** - The map is imported when the value is True. (boolean)
- **init_pose_x, init_pose_y, init_pose_phi** - The initial estimation of the robot position in the imported map. (meters, meters, radians)(double, double, double)
- **min_distance_, min_rotation_** - The scan matching is computed, when the odometry reaches the distance or rotation angle. (meters, degrees)(double, double)
- **visualize_occ_** - The map is available as the Occupancy grid when the value is True. (boolean)
- **visualize_ellipse_** - The map is available as a set of ellipses when the value is True. (boolean)
- **visualize_every_x_sec_** - The time period for the visualization. The default value is set to 3.0 seconds. (seconds) (double)

4.2.3 EllipsoidViewer

The class *EllipsoidViewer* is one of two possible types of visualization of the map. The ROS visualization marker (*visualization_msgs::Marker*) represents the ellipse. The ellipse corresponds with percent of the mass of Gaussian.

An input parameter is a pointer to the vector with NDT cells. A size of semi-axes and an angle are computed by eigenvalues of the covariance matrix. A parameter *mean* of the cell defines a center position of the ellipse. The marker is colored depending on a cell type. The cell from the CAD drawing has a different color from the cell creating by the mapping process.

The parameters of the *EllipsoidViewer* class:

- **topic_name_** - The name of the topic with an array of markers for visualization. The default value is set to *ndt_ellipse_map*. (string)

- **nstd** - The standard deviation value defines how many percent of the Gaussian is visualized. It is tabular value for probability density function. The width w and height h of the ellipse is given by formulas $w = 2 \text{ nstd } \sqrt{\lambda_1}$, $h = 2 \text{ nstd } \sqrt{\lambda_2}$, where λ_1, λ_2 are the eigenvalues of the covariance matrix. The default value is set to *3.22*, which correspond with 80% mass of the Gaussian. (double)

4.2.4 OccupancyGridViewer

The other type of visualization is the Occupancy grid. It is discretization of the probability density function into the grid with the higher resolution than the NDT grid. The input parameter is a pointer to the NDT grid. Width and height in pixels of the occupancy grid are computed by the *cell_divider* parameter, which defines the number of pixels in the NDT cell. The procedure is similar to the *EllipsoidViewer*. The parameters of the ellipses are computed from the eigenvalues of the covariance matrix. The mask, which defines the occupied cells is 'printed' into the Occupancy grid. The cell with no Gaussian or with the small value of occupancy are set as free.

The parameters of the *OccupancyGridViewer* class:

- **topic_name_** - The name of topic with the standard ROS message *nav_msgs::OccupancyGrid*. The default value is set to *ndt_occupancy_map*. (string)

- **cell_divider_** - The value defines resolution of the Occupancy grid map. Example: If the *cell_size* parameter is set to 0.25 m and *cell_divide* to 13, the size of one pixel in the Occupancy grid map is $\frac{0.25}{13} = 0,01923\ m$. The default value is set to 13. (integer)

4.2.5 P2DRegistration

The class *P2DRegistration* implements the PCL-to-NDT scan matching as described section 3.2. The implementation has been developed by Jelínek [3] based on paper [16]. The multilayer scan matching has been used in that work. The upper layers have a lower resolution than the layers on the bottom.

The scan matching is used for the continuous localization. The estimation of the robot position by the odometry tends to be good enough for the convergence of the scan matching. The one layer scan matching works properly. The size of the cell can be the same as in the NDT map or n-times bigger. The lower resolution grid is computed by the recursive covariance update (RCU). The scan registration worked better with the lower resolution in some experiments.

The input parameters are the measured point cloud, the pointer to the NDT map and the estimation of the robot position by the odometry. The output is the transformation between the point cloud and the NDT map.

The parameters of the *P2DRegistration* class:

- **score_epsilon_** - The parameter describes an absolute value of the difference between the value of the score function Eq 3.10 in the time t_i and t_{i+1}. If the difference is less than the *score_epsilon_*, the convergence criterium is reached. The default value is set to 1. (double)

- **max_iterations_** - The maximum number of iteration. If the number of iteration reaches the *max_iterations_*, the scan registration returns current transformation even though the convergence criterium has not been reached. The default value is set to 20. (integer)

- **cell_size_** - The cell size for the scan matching. The default value is the same as *cell_size_* in the NDT map. (double)

4.2.6 NdtCell

The *NdtCell* class describes the NDT cell by these parameters: the covariance matrix, mean vector, center of the cell, inverse covariance matrix, probability of occupancy, number of points in the cell, temporal vector with points in the cell and a boolean value signs that the cell comes from the CAD drawing.

The Singular Value Decomposition (SVD) of the covariance matrix is applied, and a diagonal matrix with eigenvalues on the diagonal is stored. The inverse covariance matrix is stored as well for the faster computation in the Newtons method described in section 3.2.2, even though this information is redundant. The probability of the occupancy is computed by the NDT-OM described in section 3.3.1. The number of points in the cell needs to be stored for the Recursive Covariance Update (RCU) described in section 3.3. The temporal vector with the measured points is used for computation of the mean vector and covariance matrix. The cells from the CAD drawing is not updated by the RCU, thus they are static (unchangeable over time).

The methods in this class deal with operations such as computing the mean value and the covariance matrix from the measured points, the merging two cells into the one with the RCU.

4.2.7 NdtGrid

The *NdtGrid* class handles operation with the NDT cells. This class is used for the NDT map representation. The important functionality is computing the NDT grid from the measured point cloud or registration of the point cloud to the existing NDT grid. The registration can be done with or without the ray tracing. The ray tracing is able to model the free cell and updates the occupancy probability so the dynamic object can disappear.

The point cloud, with the mean value of the cells, is stored as the k-D tree. This structure is used for the fast search for the nearest neighbor cell in the scan matching.

The parameters of the *NdtGrid* class:

- **cell_size_** - The size of cell in the NDT grid. The default value is set to 0.25 m. (meters) (double)
- **origin_** - The position of the NDT grid in the global coordinate system *fixed_frame_*. (meters, meters, radians)(3 doubles)

Chapter 5

Experiments

5.1 Experimental platform

The lidar is the crucial range sensor for the mapping and localization in this work. The experimental platform contains the Sick TiM361 lidar and the Robot Jackal. The robot is equipped with some additional sensors like a camera or an Inertial Measurement Unit (IMU). The camera data is not used in this work. The wider concept of solution for autonomous robots will use the camera for global localization.

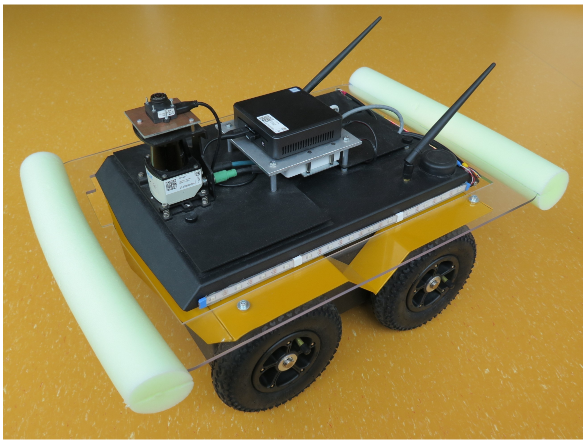


Figure 5.1: Experimental platform

5.1.1 Robot Jackal

The Robot Jackal from the Clearpath Robotics is a four-wheeled ground vehicle. The robot is designed as experimental platform for developing applications in the mobile robotics. The robot has two motors with 500 watts power. Each motor propels two wheels on the same side (four-wheel drive).

The differential kinematics allows the rotation around robot center. The 270Wh battery powers the robot for about four hours.

The robot senses data with gyroscopes and accelerometers, which create Inertial Measurement Unit (IMU). Another source of information about the robot moving is odometry. The weight of the robot is 17 kilograms, and dimensions (L x W x H) are 508 x 430 x 250 mm [17].

Two computers are used. The computer inside the robot has CPU Celeron J1800, Dual core 2.4 GHz, 2 GB RAM, 32 GB Hard Drive, operating system: Ubuntu 14.04 with the ROS Indigo). This computer is used for collecting the data from sensors and for motion control.

The second computer performs computation of SLAM. The Intel NUC computer is mounted on the top of the robot (Intel Core i7-8650U, 4.20 GHz, 8 cores, 16 GB RAM, 250 GB Hard Drive). This computer uses Ubuntu 16.04 with the ROS Kinetic as an operating system. The router connects both computers with an Ethernet interface, and there are two other free ports for connecting additional devices such as the external notebook. The NUC computer also provides WLAN for wireless communication.

The required time for one scan registration computation on the NUC computer takes around 320 ms (rarely around 780 ms) depending on the number of iteration in Newtons optimization.

5.1.2 Lidar Sick TiM361

The lidar Sick TiM361 emits laser pulses reflected by a rotating mirror. The pulses paths form a pencil of lines in the horizontal plane. The measured distances are performed with the angular resolution 0.33° [18]. The connection between the lidar and the robot is via the Ethernet interface. The technical parameters are listed in Tab. 5.1.

Sick TiM361	
Light source	850 nm
Laser class	1, eye-safe
Aperture angle (horizontal)	180°
Scanning frequency	15 Hz
Angular resolution	0.33°
Working range	0.05 - 10 m
Systematic error	±60 mm
Statistical error	20 mm
Connection	Ethernet, USB
Ambient light immunity	80 000 lx

Table 5.1: Parameters of the lidar Sick TiM361

5.2 Experiments description

The experiments are mainly focused on the influence of the imported NDT map, based on the CAD drawing, on the quality of the mapping process. The results of computation SLAM with and without using CAD drawing will be compared on the same datasets. The aim is to describe quality of the mapping process and causes of failure in localization.

The first experiment describes the common behavior of the program, differences between mapping with and without using CAD drawing. The second experiment is focused on the long-term program performance when the robot cycled twenty times around the lecture hall. The third experiment focuses on failures of the localization and mapping process.

All experiments have these common properties:

- The Robot Jackal has been guided manually with joystick (PS4 controller) in an office environment.
- The maximum speed of robot has been 0.5 ms^{-1}.
- The frequency of lidar scans has been set to 15 Hz.
- The frequency of odometry computation has been set to 50 Hz.

The NDT cells in all figures in this chapter follow these colors:

- The blue color is used for the NDT cells, which were created from CAD drawing (the imported map).
- The green color is used for the NDT cell, which were created by mapping process (from the measured point clouds).

5.2.1 First experiment

The map has been created by two passage of the robot in the environment. The program has been executed with the input dataset specified in Tab 5.2 without and with the imported map from the CAD drawing.

duration	7 min 12 s (432 s)
number of the laser scan messages	6 485
number of the odometry messages	21 616

Table 5.2: Dataset description of the first experiment.

The parameters for scan matching computation were set to (min_distance = 0.15 m, min_rotation = 3°). The other parameters were set to the default values.

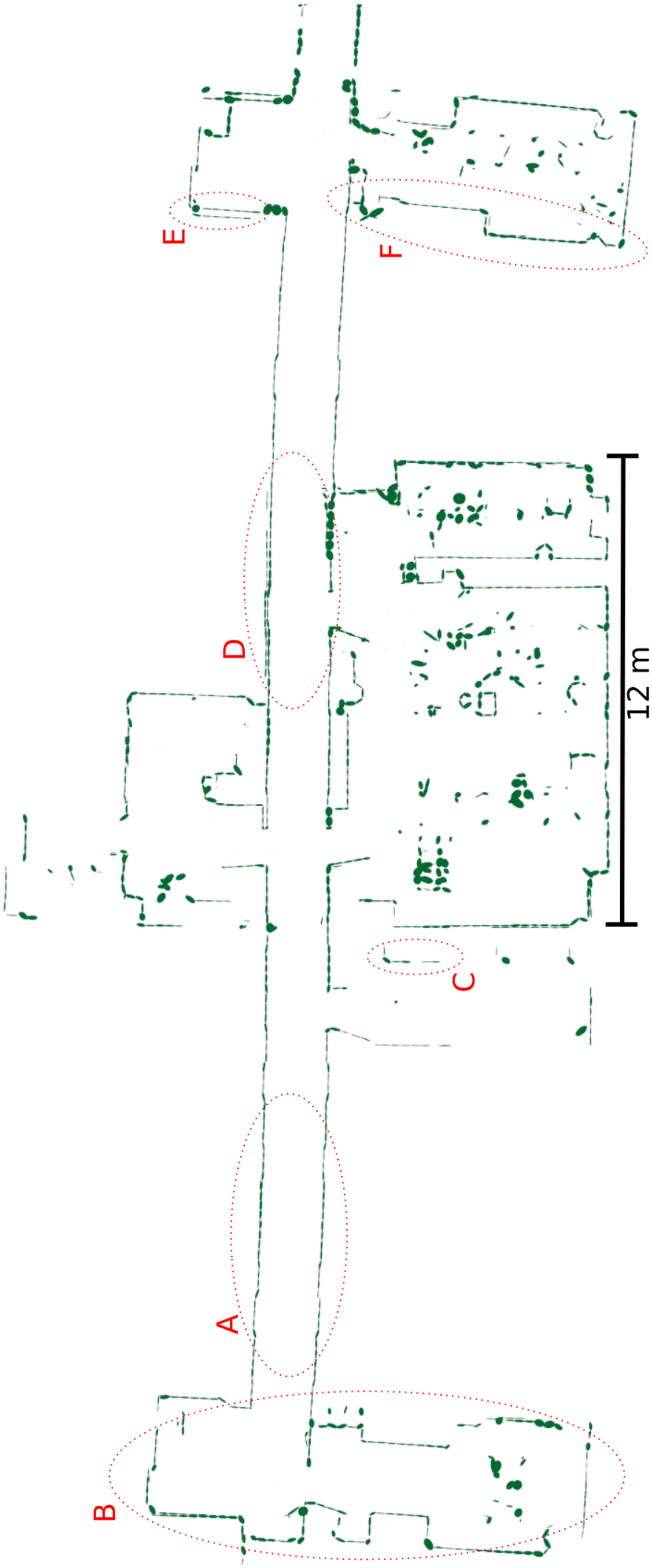


Figure 5.2: The final map of the first experiment **without** imported map from CAD drawing.

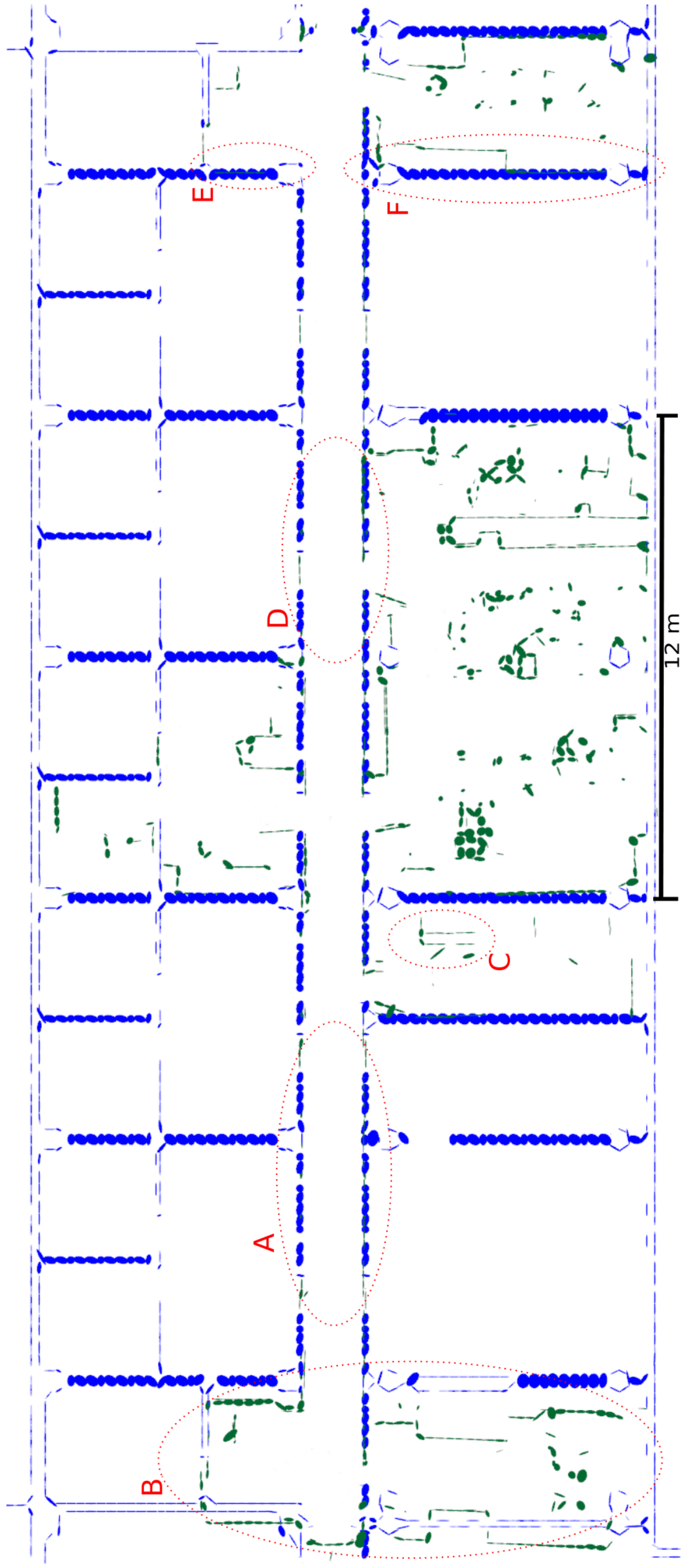


Figure 5.3: The final map of the first experiment **with** imported map from CAD drawing.

Without CAD drawing

The final map without using CAD drawing is shown in Fig 5.2. The areas marked by red dotted ellipses are analyzed below.

The area **A** corresponds to the long corridor. The scans from lidar are very similar, and it is difficult to find their correct position on the map. The scan matching registers the point cloud with the error. The estimated length of the corridor is larger than its real length.

The consequence of the error in the long corridor is the incorrect placement of the area **B**. The scan matching is consistent, and the area **C** is not duplicated when the robot rides in the opposite direction even though the length of the corridor was estimated incorrectly.

The area **D** is slightly duplicated during scan matching. This area has been observed multiple times from both sides of the corridor and from the adjacent room. The scan matching most likely neglected the width of the wall between the room and the corridor. The duplicity is probably caused by the fact, that one time the point cloud was registered to the one side of the wall and other time to the reverse side of the wall. The estimated corridor shape is warped in the map. The area **E** refers to local duplication of the map as a failure of the scan matching.

The room with the area **F** is mapped with a wrong angle. The curvature is caused by the previously warped corridor and by passing through the door. The situation when the robot rides through the door is critical. The measured scan after passing through the door contains a lot of new information (the unmapped environment) and less information about the already mapped area. The scan matching needs solid reference model for correct registration of the measured point cloud. The rotated neighboring rooms can have the overlapping walls in the map which will misguide the future localization.

With CAD drawing

The final map using CAD drawing initialization is shown in Fig 5.3. The incorrect estimated length of the long corridor in the area **A** causes failure of mapping in the area **B**. This is the same kind of failure as in the case without using CAD drawing.

A duplicity in the area **C** is created, when the robot moves in the opposite direction. The duplicity leads to the wrong localization. The scan matching is not able to converge to the global optimum and the measured point cloud is placed to the wrong position. Still, it can be seen in Fig 5.3 that misalignment in the localization in the area **C** is smaller than in the area **B**.

The scan matching in the next part of the corridor (the area **D**) finds the right position of the robot and the duplicates no more occur.

The area **F** is mapped without warping. The imported map based on CAD drawing provides a reference model of the environment for the scan matching. The imported map helps to prevent incorrect overlaps in the map.

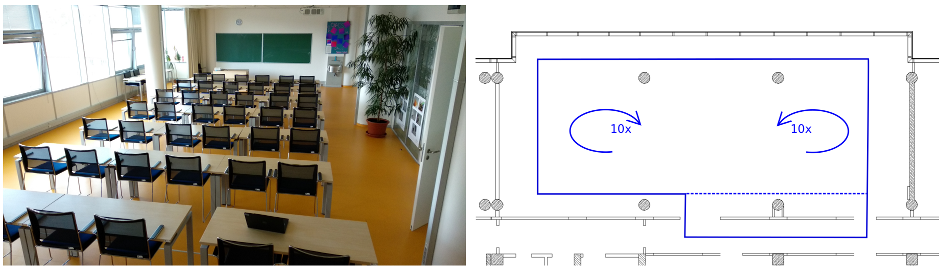
5.2.2 Second experiment

This experiment is designed for testing the evolution of the map when the robot visits mapped area multiple times. The robot rides twenty loops in a lecture hall and the part of the corridor. Ten loops in each direction have been recorded. Two boxes were placed in the corridor for preventing a failure in scan matching along the long corridor. The evolution of the map without and with CAD drawing can be seen in Fig 5.5 and Fig 5.6. The final maps are shown in Fig 5.7.

duration	35 min 48 s (2 148 s)
number of the laser scan messages	32 232
number of the odometry messages	107 438

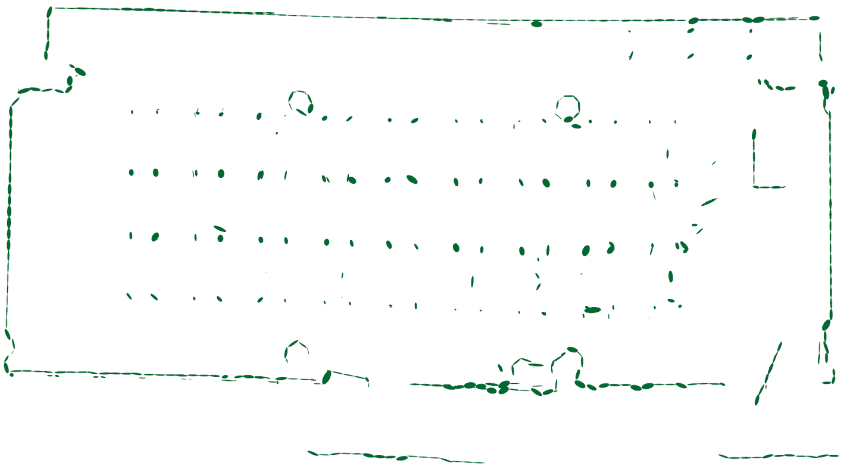
Table 5.3: Dataset description of the second experiment.

The conditions for scan matching computation were set to (min_distance = 0.40 m, min_rotation = 3°). The other parameters were set to the default value.

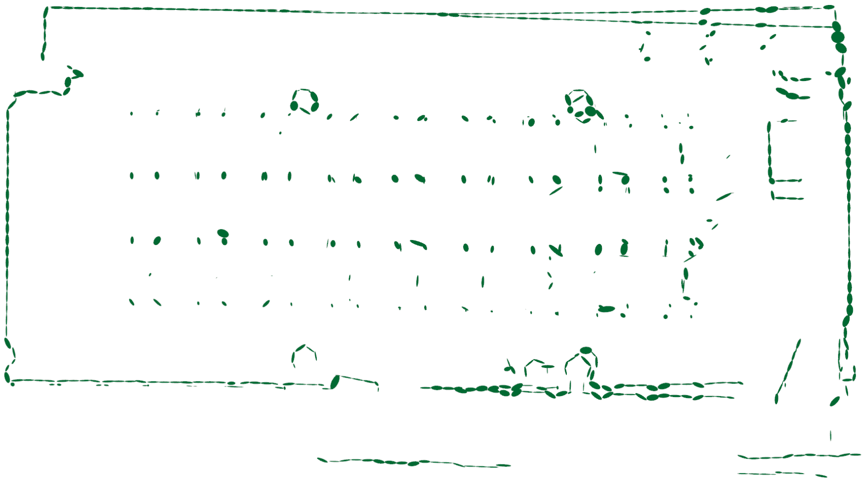


(a) : The lecture hall **(b) :** Scheme of the robot trajectory.

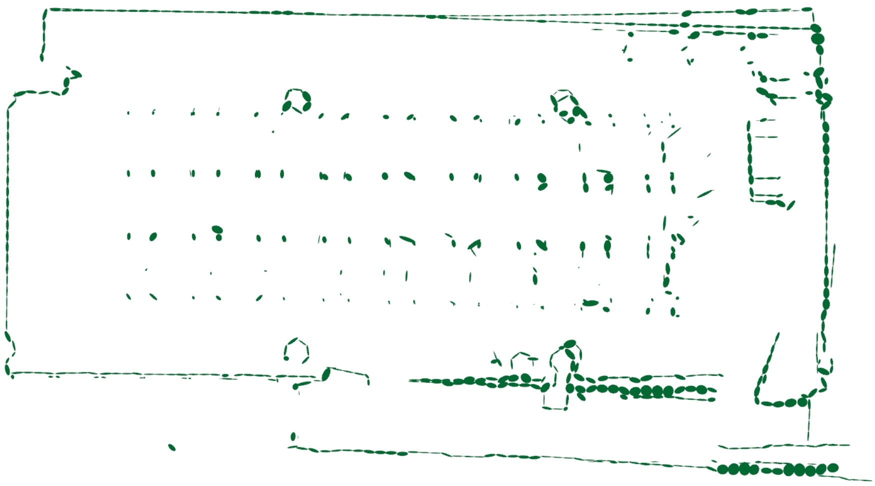
Figure 5.4: The lecture hall photo and the layout with the robot trajectory. The dotted blue line means that the robot sometimes followed this path and other times followed the path placed in the corridor.



(a) : The map after the passing of **one** loop.

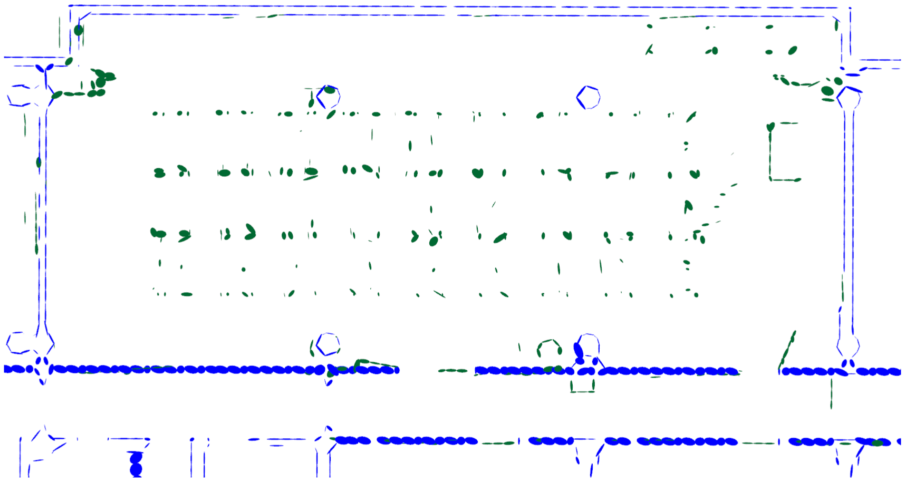


(b) : The map after the passing of **two** loops.

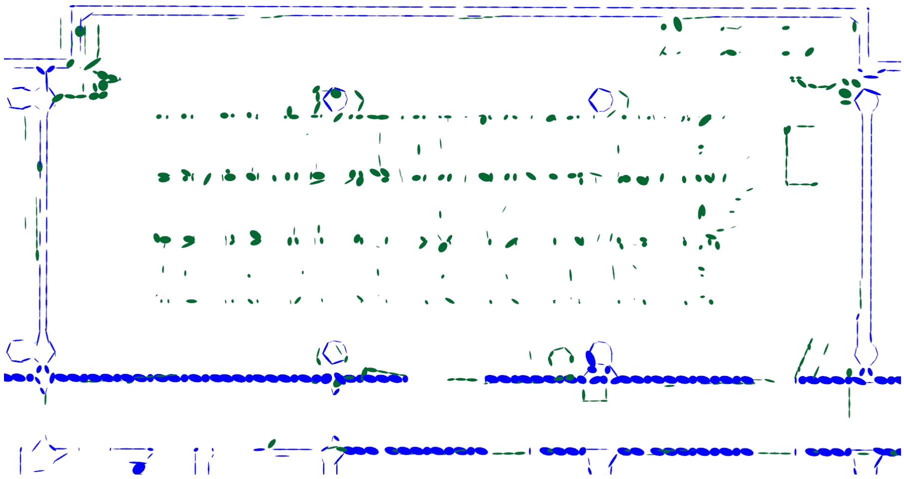


(c) : The map after the passing of **four** loops.

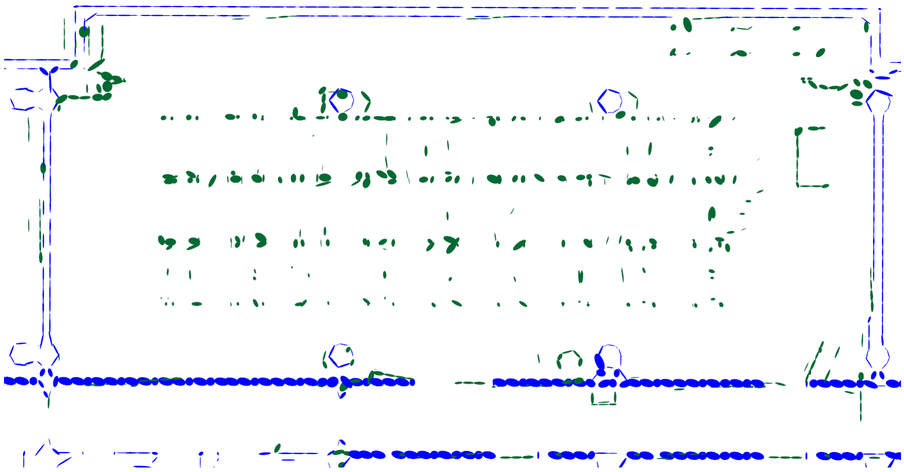
Figure 5.5: Evolution of the map in the experiment **without** using CAD drawing.



(a) : The map after the passing of **one** loop.

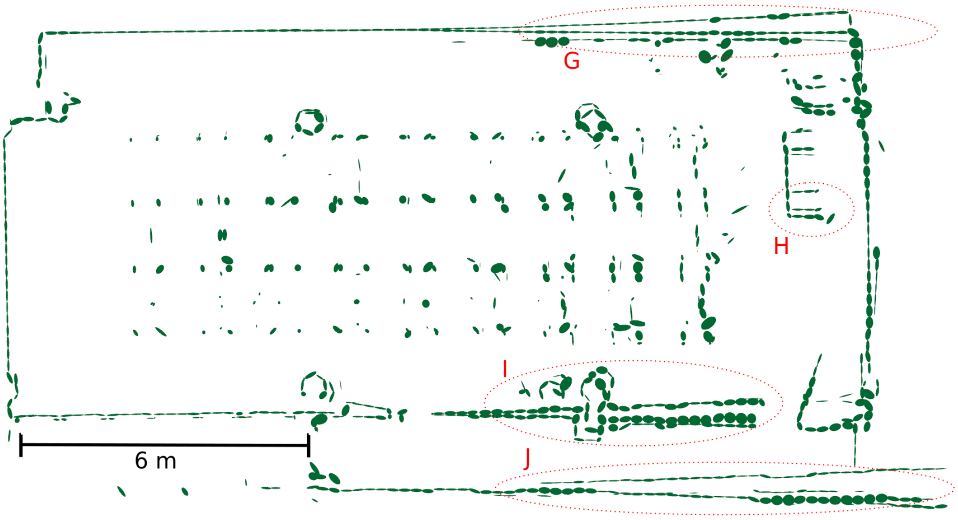


(b) : The map after the passing of **two** loops.

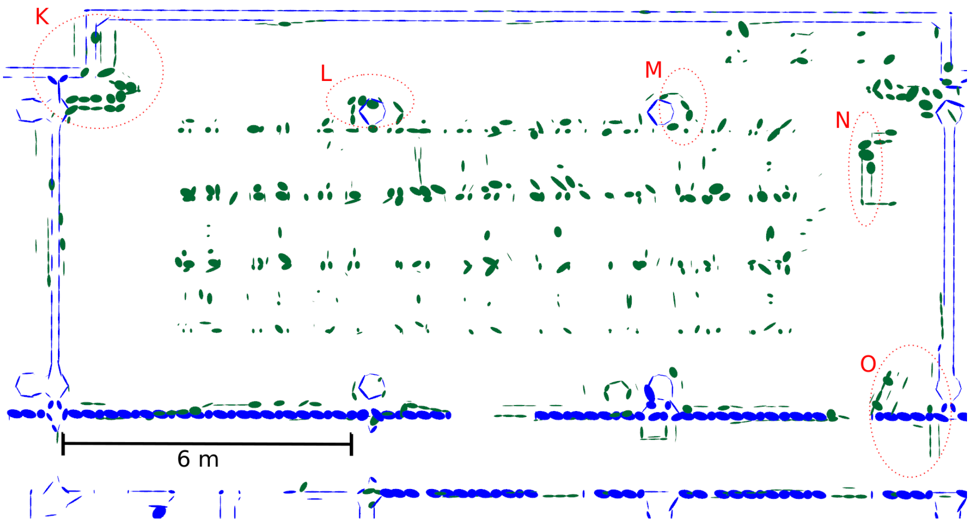


(c) : The map after the passing of **four** loops.

Figure 5.6: Evolution of the map in the experiment **with** using CAD drawing.



(a) : The final map of the second experiment **without** using CAD drawing.



(b) : The final map of the second experiment **with** using CAD drawing.

Figure 5.7: The final maps of the second experiment.

Without CAD drawing

The evolution of the map without CAD drawing is shown in Fig 5.5. The localization fails with the second pass through. The curved line in the top of Fig 5.5a is the most likely the reason. One side of the hall is duplicated during the mapping. The warping of the map is probably caused by a short range of the vision of the lidar (10 m). The length of the lecture is 18 m. The wall is modeled by many overlapping measurements, so the small errors of the individual registrations of the point cloud sum up, and the result is a curved line instead of the straight one.

The fourth loop Fig 5.5c adds another incorrect fragment into the map. The scan matching registers the point cloud to one of these three fragments in the remaining loops.

The final map without using the CAD drawing can be seen in Fig 5.7a. The areas **G, H, I, J** refer to the duplicities and the incorrect fragments in the map.

The fragment creating can be reduced by setting a larger NDT cell size. On the other hand, the large NDT cell is not able to describe the environment in the detail.

With CAD drawing

The imported map using the CAD drawing provides a reference model of the environment for the scan matching. The CAD drawing ensures straightness of the walls, so the curved line on the top of the map no more occur. The fragment is not created, and the map is more consistent. The evolution of the mapping is shown in Fig 5.6.

The final map is shown in Fig 5.7b. The areas **K, L, M, N, O** refer to the local duplicities in the map. There are surely more duplicities, but the marked ones are obvious and clearly observable. The scan matching sometimes registered the point cloud with the reverse side of the wall from the reference NDT model using the CAD. The current implementation of the method is not able to distinguish between the visible and the reverse side of the wall. Setting the NDT cells to contain both sides of the wall might be a solution.

5.2.3 Failure of the algorithm

The scan matching returns transformation between the measured point cloud and the NDT map regardless of the correctness of the transformation. The failure of the scan matching leads to the failure of the localization and mapping in general. The example of the failure is shown in Fig 5.8.

cell_size in the map	0.25 (m)
cell_size for scan matching	0.5 (m)
min_distance	0.2 (m)
min_angle	15 (deg)

Table 5.4: Parameters when program failed.

The grid-based sampling has been tested in this experiment. In the paper [16], the grid-based sampling is used for the faster scan matching. The number of points is reduced, the computation of the score function, the gradient, and the Hessian matrix in the Newtons method takes less time. The measured point cloud is separated into the grid with 0.125 m cell size and one random point from each cell is selected so the point cloud with reduced number of point is created as the input for the scan matching.

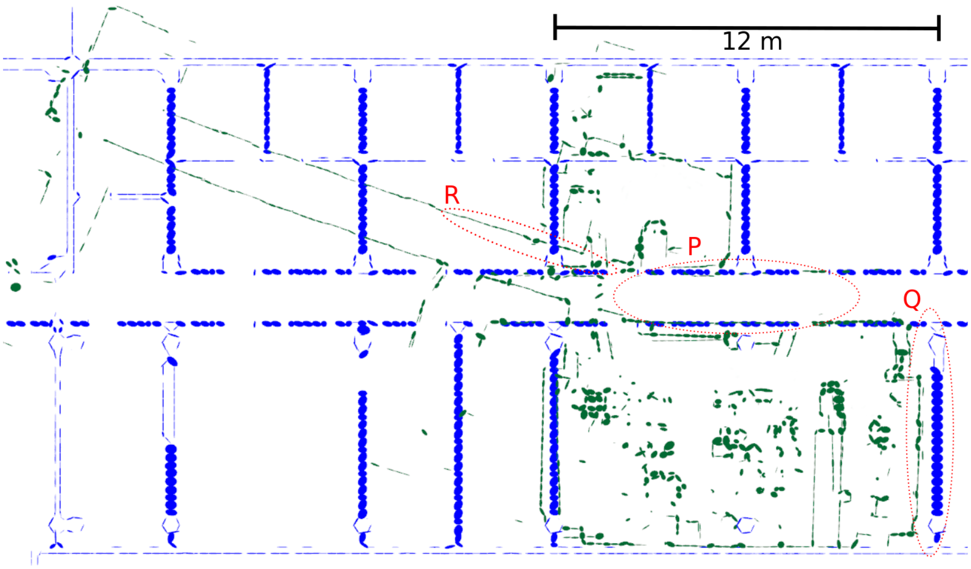


Figure 5.8: Example of the failure.

The localization fails in the area **P**. The rest of the map is shifted as it can be seen in the area **Q**. The other failure of the scan matching is in the area **R**, where the corridor is registered with the wrong orientation.

The scan matching tends to register the current point cloud with the recent mapped area. The mapped area from recent history is similar to the current point cloud. The minimum of the score function Eq 3.10 can be located in the wrong position. The mapping of the corridor keeps the wrong orientation and the reference model using the CAD drawing loses the influence for the scan matching.

5.3 Practical observations

The setting of the size of cell of the NDT map depends on the environment where the robot operates. The evaluated experiments have been done in the office environment (labs, corridors, kitchens, lecture halls). The small object such as a foot from a chair is commonly observed. These object require smaller NDT cell in the map for the sufficient descriptiveness. On the other hand, when the cell is small, the scan matching creates an incorrect fragment of the map more often. The size of cell depends on the measurement accuracy (i.e., the parameters of the lidar). This parameter needs to be adjusted experimentally. The cell size in the office environment has been set to 0.25 m

The conditions for the scan matching computation in the *AlgorithmNdtSlam* class (*min_distance, min_rotation*) have a huge influence on the behavior of the program. The small values (*min_distance = 0.03 m, min_rotation* = 1°) cause that the current point cloud is registered with a recent mapped area in the map. The original small error grows, and the process of the localization and mapping fails. The estimation of the position from the odometry determines the convergence area of the score function Eq 3.10 and it might causes convergence to the local minimum, where the method gets stuck.

The grid-based sampling has been tested as is suggested in [16]. This speeds up the registration of the point cloud from 242-310 ms to 15-45 ms. On the other hand, the reliability has been reduced, and the failures have occurred more often. The grid-based sampling is not used in the program for this reason, and all points from the measured point cloud are used for the scan matching.

5.4 Comparison of map with the Gmapping package

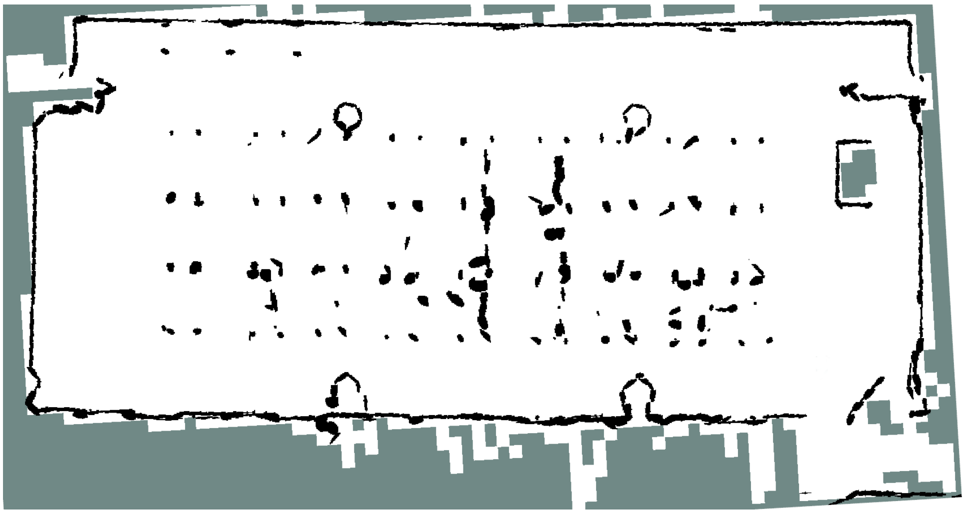
The Gmapping is the popular package for solving SLAM in the ROS. The Gmapping is based on the Rao-Blackwellized filter in which each particle carries an individual map of the environment [19]. The Gmapping package was tested on our experimental platform. The comparison of the Occupancy maps from the Gmapping package and NDT approach (our) is described below.

The Occupancy grid map was visualized in the RVIZ tool in the ROS. The figures Fig 5.9 and Fig 5.10 use following colors. The white pixels refers to the free space, the black pixels are the measured obstacles, and the green-gray pixels are the unobserved space.

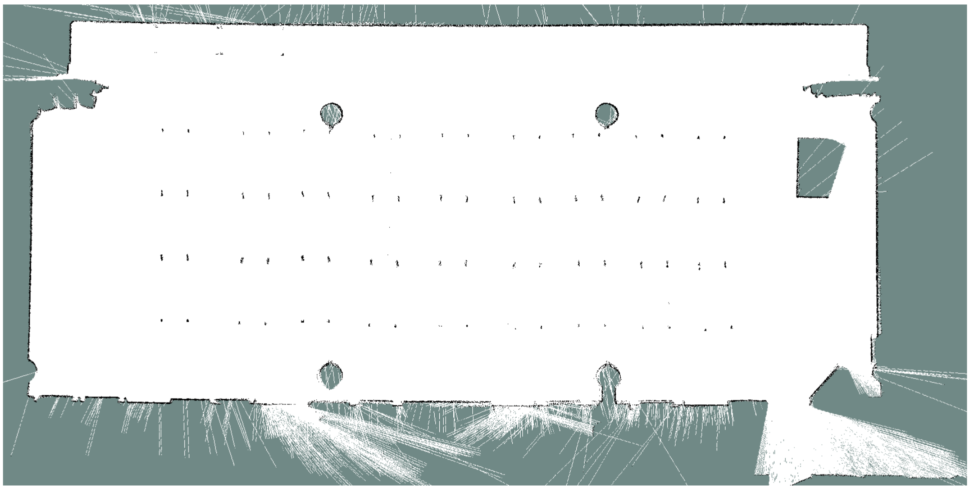
The size of one pixel in the map from Gmapping package is 0.05 m, and from NDT package is 0.0192 m. The final maps from two datasets are presented in Fig 5.9 and Fig 5.10. The Gmapping package describes the environment with more details even though the NDT package has a finer resolution of the Occupancy map. The Occupancy map computation by NDT package is done by discretization of the Gaussian in the NDT cell, so the details of the environment depend on the size of the NDT cell.

The beam from lidar can be reflected from more than one obstacle, so the measured is incorrect. The ray tracing causes that the pixels behind the objects are modeled as free space even though space has not been observed in reality. This phenomenon can be seen in each of the final maps regardless of the package.

The maps computed by the Gmapping package in Fig 5.10b is more straight in the corridors than the map computed by the NDT package in Fig 5.10a.

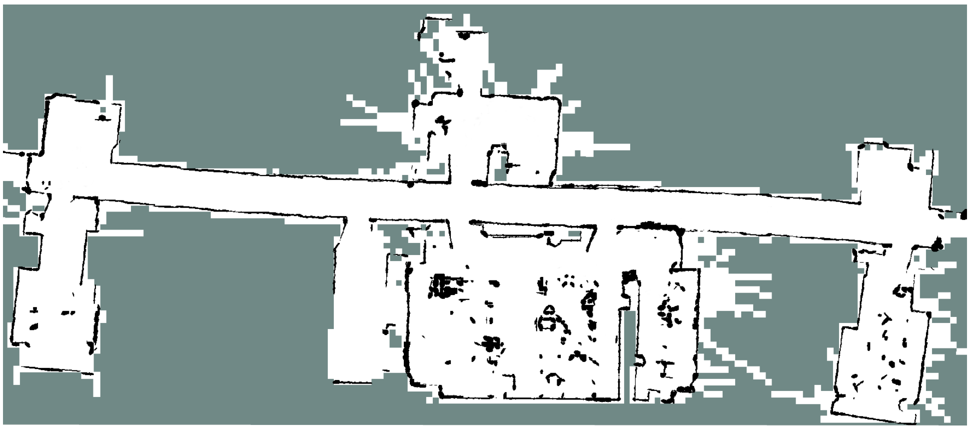


(a) : The Occupancy grid map of the lecture hall created by NDT approach.



(b) : The Occupancy grid map of the lecture hall created by the Gmapping package.

Figure 5.9: The Occupancy grid maps of the lecture hall.



(a) : The Occupancy grid map created by NDT approach.

(b) : The Occupancy grid map created by the Gmapping package.

Figure 5.10: The Occupancy grid maps.

Chapter 6

Conclusion

This work has dealt with on-line 2D SLAM solved by Normal Distribution Transform (NDT) approach. The solution is based on the incremental scan matching. The implementation takes advantage of using prior information coming from the CAD drawing provided by Pánek [4].

The implementation uses ROS Kinetic environment and follows the ROS standard representations of the robot position and the map. The program has been tested on the Robot Jackal with the lidar Sick TiM361. The results of the experiment have demonstrated that the prior information about the environment as the CAD drawing can be used as an initialized map for the localization and mapping process. The imported map from the CAD drawing improves results by eliminating the warping in the map and suppress the appearing of duplicities in the map.

Bibliography

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics.* Cambridge, Mass.: MIT Press, 2005.

[2] M. Magnusson, *The three-dimensional normal-distributions transform : an efficient representation for registration, surface analysis, and loop detection.* PhD thesis, Örebro University, School of Science and Technology, 2009.

[3] L. Jelínek, "Graph-based SLAM on normal distributions transform occupancy map," 2016. Department of Theoretical Computer Science and Mathematical Logic, Faculty of mathematics and physics, Charles University.

[4] V. Pánek, "Map import for mobile robot from CAD drawing," 2018. Department of Cybernetics, Czech Technical University in Prague.

[5] M. Kulich, "Lokalizace a tvorba modelu prostředí v inteligentní robotice," 2003. Czech Technical University in Prague.

[6] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (pcl," in *In Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 1–4, IEEE.

[7] F. Lu and E. Milios, "Robot pose estimation in unknown environments by matching 2d range scans," *J. Intell. Robotics Syst.*, vol. 18, pp. 249–275, Mar. 1997.

[8] D. Meyer-Delius, M. Beinhofer, and W. Burgard, "Occupancy grid models for robot mapping in changing environments," in *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, AAAI'12, pp. 2024–2030, AAAI Press, 2012.

[9] P. Biber and W. Straßer, "The normal distributions transform: A new approach to laser scan matching," vol. 3, pp. 2743 – 2748 vol.3, 11 2003.

[10] "M. kulich, SLAM lecture of the mobile and and collective robotics." https://cw.fel.cvut.cz/b181/courses/b3m33mkr/start. cit. 2019-05-22.

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

[12] P. Biber, S. Fleck, and W. Strasser, "A probabilistic framework for robust and accurate matching of point clouds," in *Pattern Recognition* (C. E. Rasmussen, H. H. Bülthoff, B. Schölkopf, and M. A. Giese, eds.), (Berlin, Heidelberg), pp. 480–487, Springer Berlin Heidelberg, 2004.

[13] 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," *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1627–1644, 2013.

[14] "Distributions - ROS wiki. documentation - ROS wiki [online]. avaiable from:." http://wiki.ros.org/Distributions, 2019-05-10. cit. 2019-05-22.

[15] B. Jacob and G. Guennebaund, "Eigen 3.3." http://eigen.tuxfamily.org. cit. 2019-05-22.

[16] C. Ulas and H. Temeltas, "3D multi-layered normal distribution transform for fast and long range scan matching," *Journal of Intelligent & Robotic Systems*, vol. 71, 07 2013.

[17] "Jackal UGV - small weatherproof robot - Clearpath. Clearpath Robotics: Autonomous mobile robots [online]. avaiable from:." https://www.clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/. cit. 2019-05-22.

[18] "Tim361-2134101 | detection and ranging solutions | sick. 301 moved permanently [online]. avaiable from:." https://www.sick.com/cz/en/detection-and-ranging-solutions/2d-lidar-sensors/tim3xx/tim361-2134101/p/p369447. cit. 2019-05-22.

[19] G. Grisetti, C. Stachniss, and W. Burgard, "Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters," *IEEE Transactions on Robotics*, vol. 23, pp. 34–46, Feb 2007.

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Nováček David** Personal ID number: **435016**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

Branch of study: **Robotics**

II. Master’s thesis details

Master’s thesis title in English:

Localization of Mobile Robot Using Multiple Sensors

Master’s thesis title in Czech:

Určování polohy mobilního robotu na základě informací z různých senzorů

Guidelines:

1. Study localization and mapping (SLAM) methods for mobile robots for their lifelong operation.
2. Implement the method based on Normal Distribution Transform.
3. Test the method on the mobile robot equiped with lidar, odometry and possibly IMU.
4. Evaluate the results, make conclusions, propose improvements.

Bibliography / sources:

[1] Lifelong localization in changing environments, Gian Diego Tipaldi, Daniel Meyer-Delius, Wolfram Burgard, IJRR 2013
[2] Einhorn, E.; Gross, H.-M., "Generic 2D/3D SLAM with NDT maps for lifelong application," in Mobile Robots (ECMR), 2013 European Conference on , vol., no., pp.240-247, 25-27 Sept. 2013
[3] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard: Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters, IEEE TRANSACTIONS ON ROBOTICS, VOL. 23, NO. 1, FEBRUARY 2007
[4] Michael Montemerlo, Sebastian Thrun, Daphne Koller and Ben Wegbreit: FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem, AAAI-02 Proceedings. 2002
[5] Simon J. Julier: An Empirical Study into the Use of Chernoff Information for Robust, Distributed Fusion of Gaussian Mixture Models, 2006

Name and workplace of master’s thesis supervisor:

Ing. Vladimír Smutný, Ph.D., Robotic Perception, CIIRC

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **18.10.2018** Deadline for master's thesis submission: **24.05.2019**

Assignment valid until: **30.09.2020**

Ing. Vladimír Smutný, Ph.D. Supervisor’s signature	doc. Ing. Tomáš Svoboda, Ph.D. Head of department’s signature	prof. Ing. Pavel Ripka, CSc. Dean’s signature

III. Assignment receipt

The student acknowledges that the master’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master’s thesis, the author must state the names of consultants and include a list of references.

.

Date of assignment receipt Student’s signature

Appendix A

Structure of CD

```
ndt_ciirc_slam
    CMakeLists.txt
    include
        ndt_ciirc_slam
            algorithm_slam.h
            cell_grid.h
            ellipsoid_viewer.h
            ndt_cell.h
            ndt_grid.h
            ndt_slam_node.h
            occ_viewer.h
            point2distro_registration.h
            utils
    launch
        nuc_live_demo.launch
        ciicr_slam_testing.launch
    maps
        ndt_map.txt
    package.xml
    src
        ndt_slam_node.cpp
thesis.pdf
```