Bachelor Project

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Lifelong Localization of Mobile Robot

Matěj Kapošváry

Supervisor: Ing. Vladimír Smutný, Ph.D.
Field of study: Cybernetics and Robotics
Subfield: Robotics
May 2016

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

BACHELOR PROJECT ASSIGNMENT

Student: Matěj K a p o š v á r y

Study programme: Cybernetics and Robotics

Specialisation: Robotics

Title of Bachelor Project: Lifelong Localization of Mobile Robot

Guidelines:

1. Compile state of the art techniques of simultaneous localization and mapping (SLAM) of mobile robot.
2. Measure the real life data from SICK lidar while the environment is changing.
3. Design the method able to localize the robot from the measured data.
4. Implement the proposed method, evaluate results and make conclussions.

Bibliography/Sources:

[1] Gian Diego Tipaldi, Daniel Meyer-Delius,Wolfram Burgard; Lifelong localization in changing environments, 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.

Bachelor Project Supervisor: Ing. Vladimír Smutný, Ph.D.

Valid until: the end of the summer semester of academic year 2016/2017

L.S.

prof. Dr. Ing. Jan Kybic
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, December 15, 2015

Acknowledgements

I would like to express my sincere thanks to my supervisor Vladimír Smutný for his valuable advice, regular and constructive feedback and positive attitude. I would also like to thank other people from the Faculty of Electrical Engineering who helped me and the CIIRC for lending the hardware. Special appreciation goes to my family and close ones, who have supported me during my whole studies.

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 theses.

Prague, May 25, 2016

....
signature

Abstract

The inspiration for this bachelor thesis is an industrial customer who wants to equip his factory with unmanned ground vehicles that will transport material within the factory. Lifelong localization is a part of the field of mobile robotics that addresses this challenge, as it deals with a mobile robot that has to operate in an environment that dynamically changes over time. The algorithm has to be able not only to deal with the dynamics, but at the same time it has to work online and keep the memory used in certain bounds.

The goal of this thesis is to make the future implementation easier by providing a solid overview of the topic, bearing the industrial application in mind. Besides the research of the literature, several experiments both on the hardware and software side were conducted.

The results of the experiments allow us to evaluate the hardware and software that could be used and the outcome of the literature suggests approaches to follow. Altogether, the thesis addresses the key factors of lifelong localization and summarizes them, so they could be used easily for the future industrial application.

Keywords: lifelong localization, persistent localization, lifelong SLAM, collective SLAM, UGV

Supervisor: Ing. Vladimír Smutný, Ph.D.

Abstrakt

Inspirací pro tuto bakalářskou práci je zákazník z průmyslu, který plánuje vybavit svou továrnu průmyslovými vozíky určenými k přepravování materiálu uvnitř továrny. Celoživotní určování polohy mobilního robota je část mobilní robotiky, která řeší tento problém, neboť se zabývá činností mobilního robota v dynamicky se měnícím prostředí. Algoritmus celoživotního určování polohy musí nejen umět pracovat s dynamikou prostředí, ale zároveň musí pracovat online a s omezeným využitím paměti.

Cílem této práce je poskytnout solidní shrnutí tématu celoživotní lokalizace mobilního robota za účelem usnadnění budoucí implementace v továrně. Vedle průzkumu literatury bylo provedeno několik experimentů jak s hardwarem tak se softwarem, opět v kontextu budoucí průmyslové aplikace.

Výsledky experimentů nám umožňují zhodnotit vhodnost použití specifického hardwaru a softwaru, výstup ze studia literatury pak navrhuje směry, kterými by se mohla budoucí aplikace vydat. Celkově se práce zaměřuje na klíčové faktory celoživotního určování polohy mobilního robota a shrnuje je tak, aby byly snadno použitelné pro budoucí implementaci průmyslových vozíků v továrně.

Klíčová slova: celoživotní lokalizace, celoživotní současná lokalizace a mapování, kolektivní lokalizace, průmyslové vozíky

Překlad názvu: Celoživotní určování polohy mobilního robotu

Contents

1 Introduction **1**

2 Unmanned Ground Vehicles in Industry **3**
2.1 Our Application . . . 3
2.2 Sensing the Environment . . . 4
2.2.1 Classification . . . 4
2.2.2 Our Choice . . . 6
2.2.3 Experiment: SICK S300 . . . 6
2.3 Safety Standards . . . 7

3 Localization of Mobile Robot **9**
3.1 The SLAM Problem . . . 9
3.1.1 Mathematical Formulation . . . 9
3.2 Dynamics of Environment . . . 11
3.2.1 Representation of Dynamic Objects . . . 11
3.3 Lifelong Localization . . . 12
3.3.1 Mathematical Formulation . . . 13
3.3.2 Initial Map and Our Application . . . 13
3.4 Main SLAM Approaches . . . 14
3.4.1 Extended Kalman Filters (EKF) . . . 14
3.4.2 Particle Methods . . . 15
3.4.3 Graph-based SLAM . . . 15
3.5 Convex and Nonlinear Optimization . . . 17

4 State of the Art Techniques **19**
4.1 Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters . . . 19
4.1.1 RBPF-based SLAM . . . 20
4.1.2 Proposal Distribution . . . 20
4.1.3 Adaptive Resampling Technique . . . 21
4.1.4 Algorithm . . . 21
4.1.5 Experiments . . . 22
4.1.6 Conclusion . . . 22
4.2 Generic NDT Mapping in Dynamic Environments and Its Application for Lifelong SLAM . . . 22
4.2.1 Generic Mapping Approach . . . 23
4.2.2 Graph-based SLAM . . . 25
4.2.3 Experiments . . . 27
4.2.4 Conclusion . . . 27
4.3 Other Approaches . . . 27
4.3.1 Biologicaly Inspired SLAM . . . 28
4.3.2 Prediction-based SLAM . . . 29
4.3.3 tinySLAM . . . 29

5 Collective SLAM **31**
5.1 Our Application Versus Literature 31
5.2 Mathematical Formulation . . . 32
5.3 Main Challenges . . . 33
5.3.1 Initial and Relative Positions 34
5.3.2 Map Merging and Fusion . . . 34
5.3.3 Communication Issues . . . 35
5.3.4 Dynamics of the Environment 36
5.3.5 Computational Complexity . . . 37
5.4 Approaches . . . 37

6 Experimental Setup and Experiments **39**
6.1 Hardware . . . 39
6.1.1 Pioneer 3-AT . . . 39
6.1.2 SICK LMS111 . . . 40
6.2 Software . . . 43
6.2.1 How ROS Works . . . 44
6.2.2 Used Packages . . . 45
6.3 Experimental Setup . . . 47
6.4 Experiments . . . 48
6.4.1 Measured Data Format . . . 48
6.4.2 Measuring Real-Life Data in Office Building . . . 50
6.5 SLAM Algorithms Available in ROS . . . 53
6.5.1 Gmapping . . . 53
6.5.2 Hector_slam . . . 53
6.5.3 Stereo_slam . . . 53

7 Conclusion **55**

Bibliography **57**

A Contents of Enclosed CD **61**

Figures

3.1 SLAM process as dynamic Bayesian network 10
3.2 Dynamic occupancy grids and NDT maps 12
3.3 Proces of building a graph 16

4.1 Hybrid NDT occupancy map . . . 23
4.2 Process of mapping 25

5.1 Two-robot SLAM process as dynamic Bayesian network 33
5.2 Various ways of map merging . . 34

6.1 Mobile robot Pioneer 3-AT 39
6.2 Laser range finder SICK LMS111 40
6.3 LMS 111: operating range diagram 41
6.4 LMS111: operating principle . . . 42
6.5 LMS111: beam diameter 43
6.6 The communication within the ROS structure 45
6.7 The setting of our experiment . . 47
6.8 Photo of our experimental setting 48
6.9 Structure of odometry data 49
6.10 Structure of laser data 49
6.11 Print screen of visualization of laser data in *rviz* 51
6.12 The position in which the Figure 6.11 was taken 51
6.13 Example of odometry data published at */RosAria/pose* topic . 52
6.14 Example of laser data published at */scan* topic 52

Tables

2.1 RS-422 telegram structure for continuous data output 7

6.1 The technical specification of LMS111-10100 41

Chapter 1

Introduction

The world of mobile robotics is extremely dependent on the robots' ability to localize, navigate and map in an environment, as only then they can be truly autonomous. This has been a research topic for a few decades. The problem of lifelong localization addresses the ability of a mobile robot to work in an environment that dynamically changes over time, which is basically any real-world environment. At the same time, the algorithm has to work online and preserve the memory used.

An example of such a dynamic environment can be a factory. The inspiration for this thesis is an industrial customer who wants to equip their factory with unmanned ground vehicles (UGV) that will transport material within it. Implementing such a system requires understanding the challenge of lifelong localization of a mobile robot.

The goal of this thesis is to make the future implementation of the UGVs easier by going through the available literature and suggesting approaches to follow. As the application of the thesis has to be reliable and robust, we want to rely on proven techniques and approaches and not to reinvent the wheel by building our own algorithms entirely from scratch. The objective of this thesis is to provide future researchers with a comprehensive overview they can build the solution on. In addition to theoretical research we also conducted several experiments, having examined qualities of both hardware (mainly sensors) and software that could be used.

During the work on this thesis we slightly digressed from the initial guidelines, as we had decided to put the main emphasis on the research of literature and conducting basic, yet contributing experiments.

The work is organized as follows: Chapter 2 describes the future factory application and examines key aspects of UGVs - sensors and safety standards. Chapter 3 provides a mathematical description of the problem and introduces the main methods used to deal with it. The state-of-the-art algorithms for lifelong localization suitable for our application are described in Chapter 4. As the future application of mobile robots in the factory would be more efficient if more robots were used, we devoted Chapter 5 to the topic of cooperation of several mobile robots on one task. Finally, Chapter 6 describes our experiments, including details about the hardware and software used. Chapter 7 summarizes the whole thesis and evaluates the results.

Chapter 2

Unmanned Ground Vehicles in Industry

For many years, factories and warehouses have been equipped with autonomous vehicles. Nevertheless, these self-driving vehicles are not truly autonomous, as they ordinarily use wire technologies (wires providing the vehicle with a signal, located in the floor), tapes on the ground (which have a different color than the floor, thus a light sensor can be used to follow the tape) or magnets for navigation.

This kind of navigation that relies on physical indicators tends to be failure-prone in the long-term aspect - typically the easier the installation of the indicators is, the more often problems appear. At the same time it lacks flexibility in terms of obstacle avoidance, because there are only pre-defined paths - if an obstacle appears on its trajectory, it has to stop. Moreover, crossings may cause trouble as well. Furthermore, if the arrangement in the factory changes, it may require costly changes. Here, unlike the case of failures, the easier the installation of the indicator is, the cheaper the changes are. Hence, usage of these technologies is quite double-edged.

Thus, the industrial applications today and in the future will rely on combination of various sensors that can scan the entire environment of the vehicle continuously, enabling the vehicle to localize and navigate regardless of the changes and unexpected incidents that may appear. These vehicles are then flexible and can be used in different environments.

2.1 Our Application

This bachelor thesis should help with the implementation of unmanned ground vehicles (UGV) in a factory for transporting materials between different places. The UGV will be equipped with a basic map of the factory or a simple learning process consisting of several tele-operated journeys through the factory will be used. Then, in the everyday lifelong operation in a changing environment, an omnidirectional camera will be used for sensing the environment and localization in the order of meters. A LIDAR will be responsible for localization in the order of centimeters and simultaneous navigation and mapping. The localization, navigation and mapping based on LIDAR data is the topic our thesis will focus on.

2.2 Sensing the Environment

Bearing in mind the focus of this thesis, obtaining precise data about the surroundings of our robot is essential. There are several ways for a mobile robot to sense its environment.

In this section we briefly describe various sensor types for mobile robots and their classification, as well as basic characteristics, advantages and disadvantages of using each of them. Finally, we clarify what sensors we have chosen and why.

2.2.1 Classification

There are several ways of classifying sensors and the methods of sensing the environment. The following paragraphs do not cover all sensors and methods of measurement, we only focus on some of them. One of the basic classification criteria is the emission of energy.

- **Passive sensors** do not emit any energy to get an information about the environment, they rely on suitable physical characteristics of the environment. Examples: *camera, gyroscope, odometry, compass, GPS.*
- **Active sensors** detect energy they themselves emit. Examples: *LIDAR, sonar.*

We can also divide the sensors in two categories regarding the relativity of measured data - absolute and relative sensors. However, most often both absolute and relative position measurement methods are used simultaneously since each of them has its advantages and disadvantages.

- **Absolute sensors** provide us with absolute measurement data, e. g. distance to an obstacle. *LIDAR, camera, GPS and others* are part of this group. This family of sensors can be further divided as follows.
    - **Time-of-flight (TOF)** based sensing devices send a pulse and measure the time t it takes to travel to an obstacle (where it is reflected) and back to the sensor. With the knowledge of the speed of the pulse propagation v and using the very basic physics we get the distance d:
    $$d = v\frac{t}{2}\ . \tag{2.1}$$
    There are several sensors based on this method that differ in the type of the pulse.
        - **Sonar** uses an ultrasound pulse. These sensors usually reach up to 20 m with accuracy about 40 mm. It works poorly when detecting non-monolithic objects, sloping surfaces or thin items. Still, some approaches [1] use sonar as one of their sensors.
        - **LIDAR** sends laser pulses (wavelength 600 - 1000 nm). The range can be as long as 100 meters with accuracy around 30 mm

  and resolution 0.5° to 5°. The measurement ray is very thin, thus it is quite precise. It only experiences problems with very low-reflectance materials and at the edges of objects. However, it is very expensive. Despite this fact, it is broadly used in many SLAM applications, such as [2], [3], [4].
    - Recently, the start-up Sewio Networks[1] introduced a solution using ultra-wide band radio technology that should provide its customers with precise real-time indoor location. As it received an investment of 1 M$, we expect further development of their product that could potentially be used in our application.
  - **Global positioning system** (GPS) is based on measuring distances from specialized satellites, where at least three, respectively four, satellites are necessary to determine position, respectively position and altitude. Although it has plenty of applications, it is not really well-suited for the world of mobile robotics, as the GPS signal may be low in some environments and the orientation of a robot can be determined only if the robot moves.
    - Australia-based company Locata[2] presented GPS-like new positioning technology. It is not GPS-dependent and should serve in cases where GPS is erroneous. Locata uses a network of ground-based transmitters that covers an area with strong radio signal[3].
  - **Vision** sensors provide us with a huge amount of information, yet are quite affordable. Different types of cameras (omnidirectional, stereo, depth) can for example detect obstacles, recognize paths or road signs. However, the excessive amount of data can easily overwhelm the ability of an algorithm to work online. The papers [4], [5] used cameras as one of their sensors.
  - **Landmark navigation** is based on landmarks that should be easily distinguished by the robot's sensors because of their location, size, colors or other characteristics. They can even provide the mobile robot with extra information using e.g. QR codes or barcodes. Landmarks' position is in most cases known by the robot beforehand (before the localization process). One example of an approach that uses landmarks is [5].
- **Relative sensors** give us measurement data that is relative to previous measurements or an initial value. The examples of relative sensing methods include odometry and inertial navigation.
  - **Odometry** measures wheels rotation with the use of encoders in order to estimate the relative change of position. Sometimes it

[1]http://www.sewio.net/
[2]http://www.locata.com/
[3]More about the technology to be found at http://www.locata.com/wp-content/uploads/2014/07/Locata-Technology-Brief-v8-July-2014-Final1.pdf

may work surprisingly well, nevertheless, if the surface is slippery, significant errors may occur. Typically, in all approaches where it is used, odometry does not play the primary role but usually has more of a "supportive" purpose.

- **Inertial navigation** uses gyroscopes and accelerometers which measure the velocity of rotation and acceleration. Integrating these measurements, we obtain the position. The measured value usually has an offset. Since it is being integrated, the offset error grows. Therefore, the inertial navigation is not very suitable for long-time usage.

2.2.2 Our Choice

As we have already mentioned in Section 2.1, the core of the localization process in our application will be taken care of by an omnidirectional camera and a LIDAR. These two sensors will be supported by odometry and inertial navigation.

This work focuses on the localization, navigation and mapping based on LIDAR. We have chosen this sensor because of the fact that it is used very often in similar applications and it is highly accurate. When searching for the right one, we looked at SICK products. SICK is one of the oldest and leading manufacturers of LIDARs for civil applications in the world. The company provides its customers with sensors and application solutions for safety of people and prevention of damages.

For the purpose of this thesis, the SICK LMS 111 was used, because it was available to be borrowed from the CIIRC[4]. The sensor is described in detail in the Section 6.1.2.

Some researchers (for example [4]) were able to perform SLAM using only the LIDAR. However, as the environment in the factory can be quite monolithic, the robot might experience difficulties when localizing itself, since the laser data could be insufficiently determinative. This is why we want to use also a camera in our future application.

2.2.3 Experiment: SICK S300

There are two important product groups for our application in its portfolio - security laser scanners (such as SICK S300) and 2D laser scanners (such as SICK LMS 111). At first glance, both of these groups seem to be alike - both use the time-of-flight principle of distance measurement. However, their applications are different.

Whilst the 2D laser scanners just measure their surroundings, the safety laser scanners have important security aspects. One can define "protective fields" within the surroundings of the sensor and if an object or person enters that area, a safety laser scanner stops a subsystem device that could potentially collide with an intruder.

[4]Czech Institute of Informatics, Robotics and Cybernetics, https://www.ciirc.cvut.cz/

The goal of our experiment was to find out whether we can obtain the laser measurement data from a safety laser scanner that is working simultaneously in the safety-mode, since we need this data for the localization and navigation of the mobile robot, while guaranteeing the safety function as well.

For our experiment we borrowed a SICK 30B- 3011BA from the local SICK representative. The device operating instructions [6] gave us information about the system interface. Four pins serve as the RS-422 interface for outputting measured data (RxD-, RxD+, TxD- and TxD+). Using a RS-232/RS-485 converter and a serial port to USB converter on the hardware side and Mtty[5] terminal on the software side we were able to display the raw data.

Thanks to the Telegram Listing Standard documentation [7] for the SICK 300 scanners we identified the format of the output measured data (see Table 2.1). With the help of CDS[6] the baud-rate on the RS-422 interface can be configured to the values of 9600 Baud to 500 kBaud. If the baud rate is set to 500 kBaud, the S300 sensor is able to transfer the measured data of every second scan in real time.

For our application, we need to receive data continuously. If such a mode is chosen, the telegram structure is following: 8 data bits, 1 stop bit, no parity bit.

Telegram header	Administration data	Measured data	CRC
4 bytes	6 bytes	1094 bytes	2 bytes

Table 2.1: RS-422 telegram structure for continuous data output

- Telegram header - $0x - 00000000$
- Administration data - $0x0000$ for continuous output, 2 bytes for size of telegram and 2 bytes as coordination flag and device address
- Measured data - including configurable contour
- CRC - 16 bit, formed according to $x^{16} + x^{12} + x^6 + 1$ polynomial

These findings allow us to use just the safety laser scanner in the future application. That is not only more convenient in terms of construction, mounting and wiring, but also less expensive, since we do not have to equip the vehicle with two costly sensors.

2.3 Safety Standards

Safety should be kept in mind in all robotics applications, and especially in the mobile ones. That is why we worked with the official standards: The European Standard EN 1525:1997 (Safety of industrial trucks - Driverless

[5]https://www.netburner.com/learn/serial-terminal

[6]Configuration & Diagnostic Software, provided by SICK, available online: https://www.sick.com/us/en/downloads/software.

trucks and their systems) and its Czech version ČSN EN 1525 (Bezpečnost motorových vozíků - Vozíky bez řidiče a jejich systémy, [8]). These standards deal with all vehicles that were build to move autonomously without a driver.

Primarily, the industrial trucks have to be able to detect a person in their trajectory. Several conditions are to be met, for our application the following are the most crurial ones:

- the vehicle has to be able to detect a person within at least the whole width of the vehicle in all directions of movement,
- it has to detect a cylindrical item (diameter 200 mm and length 600 mm) lying perpendicularly to the trajectory, anywhere in the environment of the vehicle,
- it has to detect a cylindrical item (diameter 70 mm and height 400 mm) standing on the ground, anywhere in the environment of the vehicle.

We have to take these rules into consideration for example when choosing the laser resolution.

Chapter 3
Localization of Mobile Robot

The localization of a mobile robot is one of the fundamental challenges in mobile robotics. The mobile robot has to know where it is before it starts performing an action. Robot localization is a problem of determining a robot's pose within a known map of an environment or determining its pose relatively to its initial position.

3.1 The SLAM Problem

According to [9], the Simultaneous Localization and Mapping (**SLAM**) is generally regarded as one of the most important problems in the pursuit of building truly autonomous mobile robots. SLAM, as its name suggests, deals with navigating a robot within an unknown environment, trying to localize the robot within the environment and mapping the environment at the same time. The complexity lies in the fact that to make a robust localization of the robot, it has to have an accurate map of its environment, however, to obtain such a map, it has to know its position correctly (a chicken-and-egg problem).

3.1.1 Mathematical Formulation

Since sensors are not noise-free, a probabilistic approach for formulating and solving the SLAM problem is common, as it can model the uncertainties formally.

The two probably most comprehensive publications of the last two decades in the field of robotics and mobile robotics [10] and [9] define the *online SLAM* as follows:

$$p(x_t, m | z_{1:t}, u_{1:t}) \tag{3.1}$$

where x_t is the robot's position at time t, m is the map of the environment, $z_{1:t}$ are measurements and $u_{1:t}$ are control inputs. The second form of SLAM problem is so called *full SLAM* - in this case the posterior is estimated over the entire robot's path:

$$p(x_{1:t}, m | z_{1:t}, u_{1:t}) \ . \tag{3.2}$$

Some papers (such as [11], [12]) also include the initial position of the robot x_0:

$$p(x_t, m|z_{1:t}, u_{1:t}, x_0) \ . \tag{3.3}$$

But since x_0 can be chosen arbitrarily [12], it does not really make a difference.

According to [12], the SLAM process and its formulation 3.3 can be modeled by a Dynamic Bayesian Network as shown in the Figure 3.1. Those elements that are represented by squares in the picture stand for observed variables $(u_1 : u_t, z_1 : z_t)$, the round ones represent hidden variables $(x_1 : x_t, m)$.

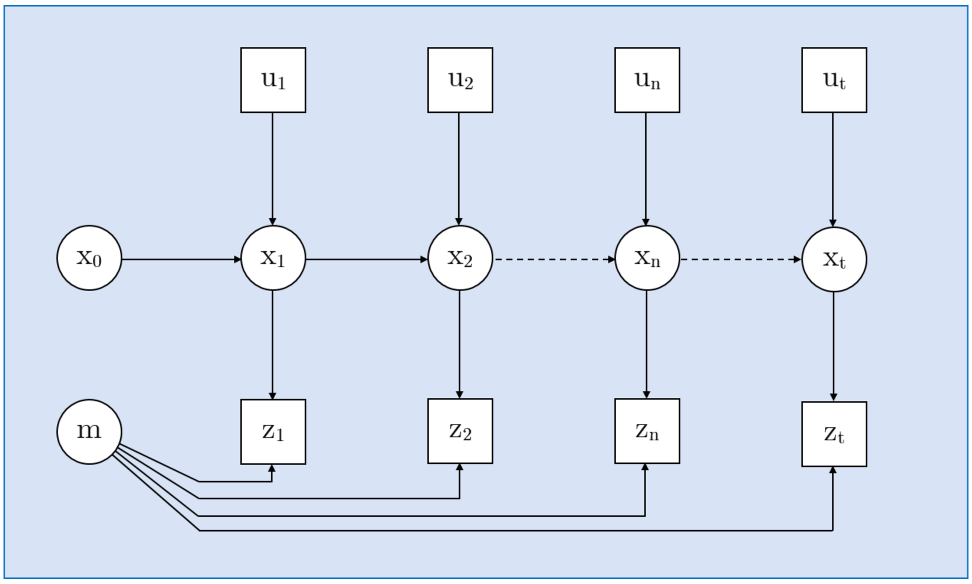


Figure 3.1: SLAM process as dynamic Bayesian network, drawn on the basis of [12, Fig. 4, p. 2]

The edges that lead to x circles represent the *transition model* (sometimes referred to also as *motion model* or *vehicle model*) that expresses the probability of a robot being at position x_t given the fact that it got there from position x_{t-1}, having measured odometry information u_t, that is:

$$p(x_t|x_{t-1}, u_t) \ . \tag{3.4}$$

The edges that lead to z squares represent the *observation model* that expresses the probability of measuring an observation z_t given the fact that robot is located at position x_t in the map m, that is:

$$p(z_t|x_t, m) \ . \tag{3.5}$$

This representation of the SLAM problem is suitable for filtering approaches [12] that handle the SLAM problem that are described in the Section 3.4.

Having defined the transition and observation model in Equations 3.4, 3.5, we can according to [11] write down the recursive equations that give the solution to the Equation 3.3:

$$p(x_t, m|z_{1:t}, u_{1:t}, x_0) = \frac{p(z_{1:t}|x_t, m)\ p(x_t, m|z_{1:t-1}, u_{1:t}, x_0)}{p(z_{1:t}|z_{1:t-1}, u_{1:t})} \tag{3.6}$$

$$p(x_t, m|z_{1:t-1}, u_{1:t}, x_0) = \int p(x_t|x_{t-1}, u_t) \times p(x_{1:t-1}, m|z_{1:t-1}, u_{1:t-1}, x_0)\mathrm{dx}_{k-1} \tag{3.7}$$

3.2 Dynamics of Environment

Most SLAM approaches deal with a static environment, despite the fact that vast majority of real-world environments are dynamic. Not all objects in a dynamic environment have to be dynamic (that is changing its position over time). In general, we can see three kinds of objects regarding their dynamics in an environment.

- **Static objects** - this group contains objects that do not change their location over time, such as columns, walls or big machines in factories.
- **Semi-static objects** - these object seldom change their location. This group includes furniture, boxes, pallets, shelves in warehouses, and so on.
- **Dynamic objects** - objects that move all the time or very often, for example cars, carts, people, goods.

Sometimes it may be hard to tell the static and semi-static objects apart (e. g. a wall can be removed). Nevertheless, it is extremely important to consider the dynamics of the environment, mainly for the localization and mapping over longer periods of time, as the omitting of the dynamics might lead to inconsistent observations (and in most cases would lead to it), ruining the process of localization and mapping.

3.2.1 Representation of Dynamic Objects

In order to be able to model the dynamic environments, it is necessary to map both the occupied space and the free space properly. If it were not possible to model the free space, it would not be possible to represent for example a removal of an obstacle.

Over time, several ways of dealing with the dynamics of the environment arose:

- treating dynamic objects as outliers,
- separating the static and the dynamic world (= building two maps),
- object-centered representation of the dynamics (this approach is based on the idea that occupancy is caused by objects),
- simultaneous representation of the environment in different times,

- feasibility grids - an approach where a dual sensor model is used to distinguish between static and moving objects,
- mapping using experiences - environment is represented by a set of experiences, where each experience is a sequence of observations connected by visual odometry.

However, according to the research of the state-of-the-art approaches, the most effective methods for modeling the dynamics are the following.

Dynamic Occupancy Grids. Presented by Meyer-Delius et al., 2012, the dynamic occupancy grid approach is a generalization of occupancy grids (or voxels in case of 3D environments) that expects the environment to change in time. The occupancy grids (or voxels) are small squares (cubes), into which the area is divided. Each square (cube) is either occupied or free. The dynamic occupancy grids are depicted in the middle part of the Figure 3.2. This mapping approach was used in [3]. It is considered a convenient way of representing an environment for local navigation and obstacle avoidance by [13].

NDT Maps. Normal distribution transformation (NDT) maps work with an environment that is, similarly as in the case of dynamic occupancy grids, divided into voxels. The difference is, that each voxel is represented not by one of two possible values (occupied, free), but by a normal distribution, which provides more information. For an illustration of NDT map see the right part of Figure 3.2. Einhorn & Gross [4] used a hybrid form of NDT maps for their approach.

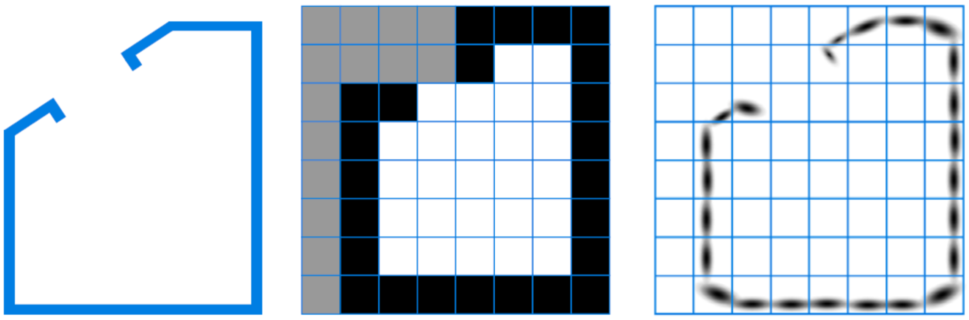


Figure 3.2: The left picture represents the structure of an environment. The middle picture shows the environment from the left picture mapped by Dynamic Occupancy Grids, where the black squares stand for occupied ones and the white ones for the free ones. The picture in the right shows the NDT mapping approach of the first picture. Figure from [4, Fig. 1, p. 29]

3.3 Lifelong Localization

We speak about lifelong localization when a robot works in an environment for a longer period of time, over which the environment changes. In order to localize itself and map the environment correctly, it has to continuously take into account the changes that occur. The following terms are also used when speaking about lifelong localization - *lifelong SLAM*, *persistent localization*.

As mentioned before, lifelong localization is way more complex than localization in a static environment. A lifelong SLAM algorithm has to be able to continuously update the map of the environment, having preserved the memory used and the computational time required. Furthermore, for vast majority of applications, the algorithm has to work online.

3.3.1 Mathematical Formulation

Tipaldi et al. [3] defined the lifelong SLAM problem in this way:

$$p(x_{1:t}, m_t | z_{1:t}, u_{1:t-1}, m_0, x_0) \tag{3.8}$$

where x_t is the robot's position at time t, m_t is the map configuration at time t, m_0 stands for lifelong map, x_0 for initial position and $z_{1:t}$ and $u_{1:t}$ again for measurements and controls.

3.3.2 Initial Map and Our Application

In our industrial application we assume that we will always provide the robot with an initial map of the environment. Not because the mobile robot would not be able to create such a map, but because the map provides a higher level of "user-friendliness" for an operator in the factory. Once an initial map is available, the operator can easily give the robot instructions such as where to start, where to go or even what path to choose. This is why we tend to work with an initial map M of the environment, which means that the equation for our SLAM application will have the following form (based on the Equation 3.8):

$$p(x_{1:t}, m_t | z_{1:t}, u_{1:t-1}, m_0, x_0, M)\ . \tag{3.9}$$

It is worth mentioning that the initial map M has a different structure than the maps m_t. Whilst M will most probably be a CAD-based map or another geometric representation of the environment, the nature of m_t maps will be probabilistic.

In the context of our industrial application, the Equation 3.9 represents a routine "optimal" operation where everything goes well. However, the real-life scenarios are never optimal - system breakdowns, robot running out of battery or others. After such an error the robot may appear in another location and the definition of SLAM problem (3.9) changes to

$$p(x_{1:t}, m_t | z_{1:t}, u_{1:t-1}, M, x_0, m_{t-1}) \tag{3.10}$$

where all symbols have the same meaning as above and m_{t-1} represents the last known-map. Often the new initial position x_0 may be unknown.

Similar situations happen when a robot is moved away from the factory for maintenance. In the period Δt where it is away, the environment can change significantly, as the Δt may be high. Therefore, the mobile robot has to deal with not only the big changes of locations of dynamic objects, but also with significant changes in the group of semi-static objects.

3.4 Main SLAM Approaches

In 2008, Siciliano & Khatib [9] observed three main SLAM paradigms[1] which are source of the majority of SLAM algorithms:

- Extended Kalman Filters (EKF),
- Particle Methods, and
- Graph-Based techniques.

However, thanks to deep research of the SLAM problem in past years, many new approaches have appeared and not all of them could fit in some of the above-mentioned categories. It is impossible to say that one approach is better than the other, despite [15] presented a way to compare SLAM algorithms (even with different outputs). In the following subsections, we will briefly look at the three main paradigms.

3.4.1 Extended Kalman Filters (EKF)

EKF were introduced already in mid 1980s. Together with the particle methods, EKF belong to the group of Bayes filters [16]. The estimated position of a robot and the state of the environment is represented by one state vector. Uncertainties of these approximations are stored in a covariance matrix. During the movement of the robot in the environment, both the state vector and the covariance matrix grow and are updated using the extended Kalman filter.

According to [11], the EKF approach tries to find the solution to SLAM problem by finding an adequate representation for the transition model (Equation 3.4) and observation model (Equation 3.5). Using the state-space representation with Gaussian noise is said to be the representation that is used most often. The transition model $p(x_t|x_{t-1}, u_t)$ is described in the following form:

$$x_t = f(x_{t-1}, u_t) + v_t \tag{3.11}$$

where v_t is Gaussian noise and function f represents the mobile robot's kinematics. The description of the observation model $p(z_t|x_t, m)$ has the following form:

$$z(t) = h(x_t, m) + w_t \tag{3.12}$$

[1]Some papers (such as [1], [14]) then started to follow this classification in their description of the SLAM problem.

where w_t is Gaussian noise and function f represents the observations. Once this state-space model is built, "standard EKF methods" (for example in [10, p. 48]) can be used to compute the probability distribution which is a solution to the Equation 3.3.

As the dimension of covariance matrix grows quadratically, some approaches propose dividing the map of the environment into submaps.

3.4.2 Particle Methods

Particle methods are based on particle filters - implementation of Bayes filters. In a way, each particle represents a guess of the real state of the environment [9]. Since [2], one of the state-of-the-art approaches we describe in the Chapter 4, is based on the particle methods (to be more specific, on a modification of it), we refer the reader to the Section 4.1.1. Another modification of the particle approach was used for example in [3].

3.4.3 Graph-based SLAM

Graph-based approaches model the SLAM problem in the form of a graph that consists of vertices that are connected by edges. This group of methods cover bigger environments (maps) more efficiently [14] than EKF, since the update time of the graph is constant and memory use is linear, whilst the covariance matrix of EKF SLAM grows quadratically.

During the research of literature, we found two approaches towards the graph-based SLAM. In the first one, used by [17] or [4], the vertices in the graph represent only the robot positions. In the second one, described in [9] and [12], the vertices represent both robot positions and landmarks in the environment. Since the first approach is used by one of the state-of-the-art work (Einhorn & Gross [4]) that we describe in the Section 4.2, we will here focus only on the description provided in [9] and [12].

Vertices and Edges

Vertices of the graph represent landmarks in the environment l_i and robot locations[2] x_t at time t. Edges of the graph can connect either two vertices of robot locations x_t, x_{t+1} or a landmark l_i and location x_t. In the case when two locations are connected, an information about odometry representing the movement from x_t to x_{t+1} is encoded in this edge. If the edge connects a location and a landmark it means that the landmark l_i was seen from the location x_t.

Building the Graph

The process of building a graph is illustrated in the Figure 3.3. For each step, the illustrations on the white background represent the real world and those

[2]The term "positions" is used also - this is why is the graph sometimes called "pose-graph", such as in [4].

on the light-blue (gray) background represent the graph.

1. At time $t = 1$ the robot sees from its position x_1 a landmark l_1, thus vertices x_1 and l_1 become connected with an edge representing the measurement.

2. At time $t = 2$ the robot is located in the position x_2, thus a new vertex x_2 emerges. This vertex is connected with vertex x_1 by edge representing the odometry information read during the movement. At the same time, the robot sees again the landmark l_1 and also landmarks l_2 and l_3 - the respective edges are thus constructed.

3. At time $t = 4$, after a few more steps, the robot gets through the location x_3 to the location x_4, having observed landmarks l_1, l_2, l_3, l_4 and l_5 during the whole motion.

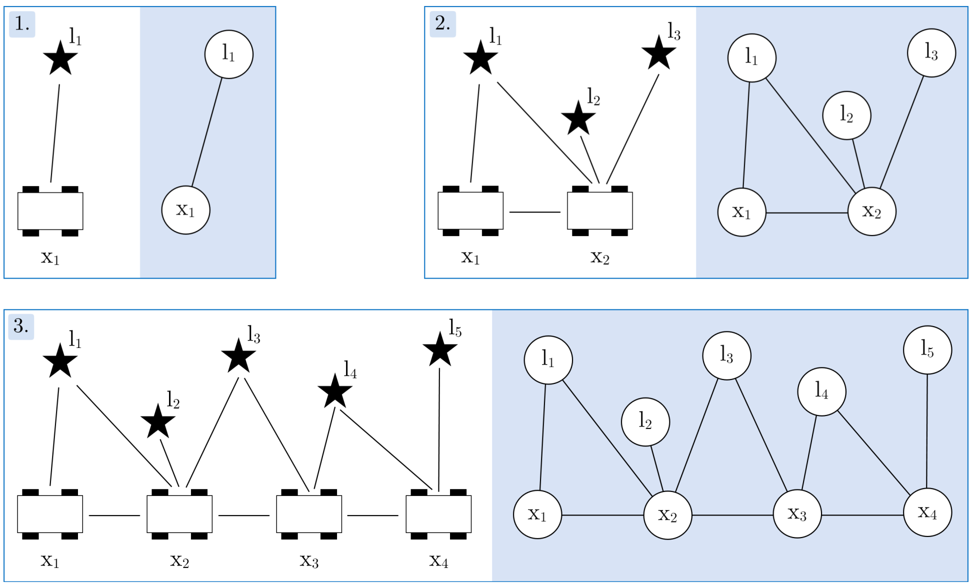


Figure 3.3: Proces of building a graph, drawn on the basis of [9, Fig. 37.4a-c, p. 878]

Matrix Representation

Since each edge of the graph connects no more than 2 vertices, we can represent the graph using matrix Ω with dimension $n \times n$, where n is the number of all vertices in the graph. In other words n is the sum of number of robot locations x_t and observed landmarks l_i, in the case of our example $n = 4 + 5 = 9$.

The elements in the matrix appear gradually, as the robot moves and explores new landmarks. If an edge exists between the i^{th} and j^{th} object in the graph, an element appears at the positions $\Omega(i, j)$ and $\Omega(j, i)$. All $\Omega(i, i)$ elements appears in the matrix as an i^{th} object appears in the graph.

For now we will only use symbols to represent the elements that appear in the matrix according to Figure 3.3: $\square$ for the first step ($t = 1$), $\triangle$ for the second step ($t = 2$) and $\Diamond$ for the third step ($t = 4$). With respect to this notation we can now look at how the matrix Ω was being created during the process in the example.

$$\Omega = \begin{array}{c|ccccccccc}
 & x_1 & x_2 & x_3 & x_4 & l_1 & l_2 & l_3 & l_4 & l_5 \\
x_1 & \square & \triangle & 0 & 0 & \square & 0 & 0 & 0 & 0 \\
x_2 & \triangle & \triangle & \Diamond & 0 & \triangle & \triangle & \triangle & 0 & 0 \\
x_3 & 0 & \Diamond & \Diamond & \Diamond & 0 & 0 & \Diamond & \Diamond & 0 \\
x_4 & 0 & 0 & \Diamond & \Diamond & 0 & 0 & 0 & \Diamond & \Diamond \\
l_1 & \square & \triangle & 0 & 0 & \square & 0 & 0 & 0 & 0 \\
l_2 & 0 & \triangle & 0 & 0 & 0 & \triangle & 0 & 0 & 0 \\
l_3 & 0 & \triangle & \Diamond & 0 & 0 & 0 & \triangle & 0 & 0 \\
l_4 & 0 & 0 & \Diamond & \Diamond & 0 & 0 & 0 & \Diamond & 0 \\
l_5 & 0 & 0 & 0 & \Diamond & 0 & 0 & 0 & 0 & \Diamond
\end{array}$$

Graph Optimization

Once a graph is built, it has to be optimized. The authors of [9] state that the Equation 3.1 can be further factorized in the following form:

$$\log p(x_t, m|z_{1:t}, u_{1:t}) = k + \sum_t \underbrace{\log p(x_t|x_{t-1}, u_{1:t})}_{\xi} + \sum_t \underbrace{\log p(z_t|x_t, m)}_{\zeta} \quad (3.13)$$

where k is a real constant, ξ represents the transition model and ζ represents the observation model.

With respect to the factorization 3.13, [9] formulates SLAM as the following optimization problem:

$$X_T^*, m^* = \underset{X_T, m}{\operatorname{argmin}} \log p(x_t, m|z_{1:t}, u_{1:t}) \ . \quad (3.14)$$

3.5 Convex and Nonlinear Optimization

A mathematical optimization problem can be defined according to [18] and [19] as follows.

Mathematical optimization. The problem of mathematical optimization (or just problem of optimization) has the following form:

$$\begin{aligned} &\text{minimize function } f(x) \\ &\text{with subject to functions } g_i(x) \le b_i, i = 1, \dots, m, \end{aligned} \quad (3.15)$$

where:

- the function $f(x) : \mathbb{R}^n \to \mathbb{R}$ is the *objective function* to be minimized over the variable x,

- the vector $x = (x_1, \dots, x_n)$ is the *optimization variable*,
- the functions $g_i(x) : \mathbb{R}^n \to \mathbb{R}, i = 1, \dots, m$ are the *inequality constraint functions* , and
- the constants $b_i, i = 1, \dots, m$ are the *bounds* for the constraints.

The solution to the optimization problem is a vector x^*, if the following condition is met: $f_0(z) \geq f_0(x^*)$ for any z where $g_1(z) \leq b_1, \dots, b_m(z) \leq b_m$.

Convex optimization. An optimization problem (as defined in 3.15) is convex, if the functions $f, g_1, \dots, g_m : \mathbb{R}^n \to \mathbb{R}$ are convex, which means they satisfy

$$h(\alpha x + \beta y) \leq \alpha h(x) + \beta h(y) \tag{3.16}$$

for all $x, y \in \mathbb{R}^n$ and all $\alpha, \beta \in \mathbb{R}$ and $\alpha + \beta = 1, \alpha \geq 0, \beta \geq 0$.

Nonlinear optimization. An optimization problem (as defined in 3.15) is nonlinear, if the functions $f, g_1, \dots, g_m : \mathbb{R}^n \to \mathbb{R}$ are not linear, but not known to be convex.

Solving convex optimization problems is easier than solving nonlinear ones, since in the case of convex functions in convex sets any local minimum is a global minimum (for proof see [19, p. 135]). We do not have any efficient way of solving nonlinear problems and even problems that seem easy usually turn into a big challenge.

In the context of the SLAM problem and application our thesis deals with: using a camera for the rough estimation (in the order of meters) of our position in the factory and *then* using a LIDAR for the more precise localization is an approach that simplifies a complex nonlinear optimization problem to a simpler convex one.

Chapter 4

State of the Art Techniques

Researchers all around the world have been dealing with the SLAM challenge for over two decades[1]. As the methods of probabilistic robotics were being developed, the research of SLAM was getting deeper and deeper. The scientists had to deal with high computational complexity and memory used. The first solutions were suitable only for offline localization and mapping. After solving the problem of being online, the ability to work in dynamic environments became one of the priorities[2].

In the following chapter we provide the readers with a selection of two state-of-the-art approaches in the lifelong SLAM problem. Having in mind the focus of this work and its future application, we focused on SLAM approaches that deal with indoor, ground and wheeled robots. The choice of the methods was based on research of literature and consultation with experts in the field.

4.1 Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters

The first technique we chose as the state-of-the-art one by Grisetti, Stachniss & Burgard [2] comes from the University of Freiburg, from the team of Wolfram Burgard, well-known and often-cited expert in the field of mobile robot navigation. He is listed as author in some other papers we worked with - [3]. What is more, this paper and also other work by these authors are source for several approaches ([20], [21], [22]) that deal with the challenge of collective SLAM, which is the topic of Chapter 5.

The Rao-Blackwellized particle filters (RBPF) are one of the particle filter approaches to the SLAM problem. In the case of RBPF each particle represents a specific map of the environment. However, the RBPF technique faces two major challenges:

1. it requires a big amount of particles to create a precise map of the environment, therefore a way to reduce it has to be found, and

[1]However, according to [11] the problem of simultaneous localization and mapping appeared for the very first time in 1986 in *IEEE Robotics and Automation Conference*. Nevertheless, the term **SLAM** was firstly used in 1995.

[2]We listed some of the approaches that may represent the dynamics of environments back in Section 3.2.1.

2. correct particles can be unintentionally removed during resampling (the literature refers to this phenomenon also as the *particle depletion problem*), consequently a method that would enable to have the ability to learn on one hand, but avoid removing correct particles on the other hand has to be introduced.

This paper deals successfully with both of them. The first problem is solved by **proposal distribution** and the second by an **adaptive resampling technique**. We will look deeper into both of these techniques in Sections 4.1.2 and 4.1.3, but firstly, let us have a closer look at how the RBPFs work in general.

4.1.1 RBPF-based SLAM

The mathematical formulation of SLAM (Equation 3.2) can be factorized in the following way:

$$p(x_{1:t}, m|z_{1:t}, u_{1:t}) = p(m|x_{1:t}, z_{1:t})\ p(x_{1:t}|z_{1:t}, u_{1:t-1})\ . \qquad (4.1)$$

The first factor represents the estimation of the map based on the trajectory of the robot, while the second factor stands for the estimation of the robot's trajectory. Because we know $x_{1:t}$ and $z_{1:t}$, the first factor can be computed analytically.[3]

An effective way to find the value of the second factor is to use a particle filter, where each particle represents a potential trajectory. The paper presents a particle-filter algorithm that works with sensors measurements and odometry data in order to update the set of samples representing the path of the robot. Briefly, the algorithm works like this:

1. **Sampling** - based on the previous generation of particles a new generation of particles is gained. The previous generation is a result of sampling from a proposal distribution π, which is in most cases based on odometry.

2. **Weighting** - each particle gets its individual weight depending on how well they suit the target distribution.

3. **Resampling** - particles are "re-drawn" according to their weights, at the end of this step they all have the same weight.

4. **Map estimation** - as we mentioned, each particle represents a potential trajectory. In this step a predicted map is computed for every particle according to formula $p(m|x_{1:t}, z_{1:t})$.

4.1.2 Proposal Distribution

In the first step of the particle-filter algorithm we were talking about a proposal distribution π. It is without surprise that the better the proposal distribution

[3]The method to construct a map based on known poses was presented in 1986 in [23].

is (that is, the more it resembles the actual distribution), the better the filter works.

Most of particle approaches use odometry motion model as the source of this proposal distribution π. However, if the robot is equipped with a sensor that is notably more precise than the model based on odometry (for example a SICK laser range finder), we can use the sensor measurement data for building the next generation of the particles efficiently. If the sensor is used, the likelihood functions are very peaked, therefore we can focus only on regions with high observation likelihood, that leads to a much more precise proposal distribution π and thus significantly less samples.

Furthermore, in most cases the likelihood functions have only one maximum (according to the authors). Consequently we can focus the sampling only on the area around this maximum and significantly reduce the computational time.

4.1.3 Adaptive Resampling Technique

In the resampling step, the particles of low importance w are exchanged for new particles of higher weight. This is important because it reduces the number of particles. However, during this reduction important samples may also be removed. Thus, a way to make a decision whether to remove a particle or not has to be found.

If all particles were representing the target distribution, they would all have the same value. In contrary, the more poorly they represent the target distribution, the higher their variance is. This is the background idea for the following decision making formula:

$$\frac{2}{\sum_{i=1}^{N}(w_i)^2} < N \tag{4.2}$$

where N is the number of particles and w_i is the weight of particle i. If the inequality is true, the resampling process starts.

4.1.4 Algorithm

The algorithm of this technique can be briefly described in the following steps, for deeper information, look at [2, p. 38].

1. Using the previous robot's pose x_{t-1} and odometry information u_{t-1} an initial guess for the robot's pose x_t^* is obtained.

2. Using the most recent map m_{t-1} a scan-matching process is executed around the initial guess x_t^*. As a result of the scan-matching we get $\widetilde{x}_t$.

3. If the scan-matching is successful, a set of sampling points is created in the region of the $\widetilde{x}_t$. These points allow us to compute the parameters of the proposal distribution.

4. The new proposal distribution is the source of the robot's new pose x_t.

5. The weights w are updated.
6. Using the measurement z_t and pose x_t, a new map is created.
7. Finally, the N (see Equation 4.2) is computed to decide whether or not the resampling will be executed.

4.1.5 Experiments

Series of experiments both in indoor and outdoor environments were conducted using several real robots equipped with SICK laser measurement systems and PLS laser rangefinders.

The experiments resulted in consistent maps with a resolution of 1 cm. The authors compare their algorithms with approaches of other groups and also demonstrate the effects of their improved proposal distribution and threshold for resampling, which are quite imposing.

4.1.6 Conclusion

The paper is very well-written and informative, as it also contains information about complexity of different operations and describes the structure of the algorithm used. Since the maps, the raw data files and the implementation of their approach are available online[4], it should be possible to use this approach in our application in the future. What is more, an open-source implementation of this technique is also available in the ROS, more information to be found in the Section 6.5.1.

4.2 Generic NDT Mapping in Dynamic Environments and Its Application for Lifelong SLAM

Einhorn & Gross presented a paper [4] that is one of the newest we have worked with as it was published in 2015. Like the first state-of-the-art technique, it comes from Germany, this time from the Ilmenau University of Technology.

The most significant contribution of this paper is that this approach is able to generate both 2D and 3D maps (without any changes to the algorithms) in a way that it does not matter what kind of sensor the robot is equipped with. This is achieved thanks to the usage of normal distribution transform mapping combined with occupancy mapping. The mapping process is further modified so it can handle free space and dynamically moving objects effectively. This advanced mapping technique is then used in a graph-based SLAM approach suitable for online and lifelong usage.

[4]https://svn.openslam.org/data/svn/gmapping/

4.2.1 Generic Mapping Approach

The authors introduced a novel approach towards mapping that combines NDT maps and occupancy grid maps in a way that they are able to model the free space. This map is called *hybrid NDT occupancy maps* and is depicted in the right part of Figure 4.1. The whole environment is partitioned into discrete voxels - *cells*. All cells are managed in a 2^d-tree[5], which makes it unnecessary to distinguish between 2D and 3D maps. Furthermore, the tree structure allows for creating maps with resolution that can be modified.

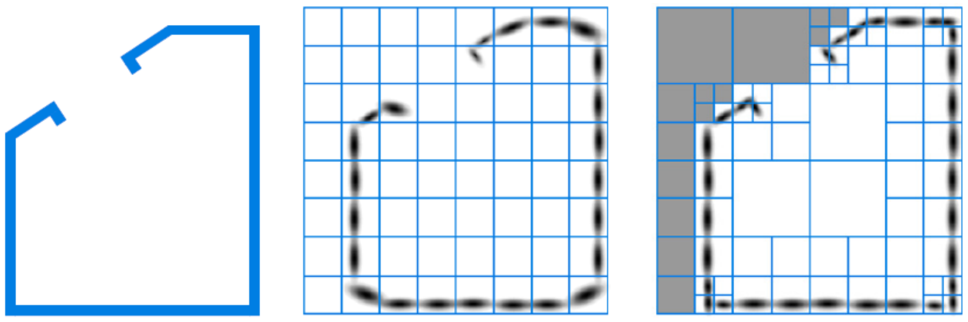


Figure 4.1: The structure of an environment in the left, NDT map in the middle and hybrid NDT occupancy map in the right, figure from [4, Fig. 1 d), p. 29]

Each of these cells is then characterized by two parameters: normal distribution $\mathcal{N}$ and occupancy value o_c. The normal distribution $\mathcal{N}(\mu_c, \Sigma_c)$, where $\mu_c \in \mathbb{R}^d$ is the mean and $\Sigma_c \in \mathbb{R}^d \times \mathbb{R}^d$ is the covariance matrix, characterizes an object covered by the corresponding cell in the environment. The occupancy value o_c represents the probability whether the cell is occupied or not - $o_c = p(c = \mathit{occupied})$. If we combine these two, we get the final probability distribution representing objects in the cell:

$$o_c \, \mathcal{N}(\mu_c, \Sigma_c) \; . \tag{4.3}$$

The above-described methods make this approach suitable for generating both 2D and 3D maps. In this paper, the independence on the type of sensor used is ensured by presenting various "sensors frontends" for different sensors[6]. All these frontends produce output data in the same format, thus the mapping process is the same for all the sensors.

Updating the Map

Initially, we assume that the state of all cells is unknown and the parameters are set as follows:

$$o_c = 0.5$$
$$\mu_c \to \infty$$

[5]The d in the upper index stands for the dimension, therefore we obtain an oct-tree structure for 3D maps and quad-tree structure for 2D maps. In the case of three (two) dimensions each cell is then subdivided into eight (four) "sub-cells".

[6]As an example, frontend for Microsoft Kinect sensor is described - [4, p. 32].

and the uniform distribution is used for the unknown (and empty) cells. The update process is run every time a new range measurement is available. Firstly, the cell that was "hit" by the measurement has to be identified. Secondly, we update the parameters μ_c, Σ_c of the Gaussian in the cell. The third task is to update the o_c.

In general, when updating the parameters, the previous values[7] μ_c(t-1), Σ_c(t-1) and the number of updates λ(t-1) are taken into consideration as the following formulas show:

$$\begin{aligned}\mu_c(\mathrm{t}) &= \alpha\mu_c(\text{t-1}) + (1-\alpha)x\\ \Sigma_c(\mathrm{t}) &= \alpha\Sigma_c(\text{t-1}) + \alpha(1-\alpha)(\mu_c(\text{t-1})-x)(\mu_c(\text{t-1})-x)^{\mathsf{T}}\\ \lambda(\mathrm{t}) &= \lambda(\text{t-1}) + 1\ ,\end{aligned}$$

where

$$\alpha = \frac{\lambda}{\lambda+1}\ . \tag{4.4}$$

The formula 4.4 for α was designed in the above-described way in order to suit the lifelong application and in order to be able to adapt to the changing environment. This approach makes sure that the newer measurements have higher weights than the older ones.

However, not only the cell which is deepest in the 2^d tree has to be updated. The update has to be recursively propagated along the ray of the sensor measurement to its parent cells.

Detection and Tracking of Moving Objects

Another major contribution of this work is the implementation of a system for detecting and tracking moving objects to their SLAM algorithm. The previous section provided us with a way to update map based on the dynamics of the environment, as the mapping approach updates the map continuously and can model free space. Nevertheless, there are still some situations when this approach may cause inconsistencies in the map.

Imagine the following situation: a robot, equipped with a sensor that has a narrow field of view, is moving in an environment. A person (or any other dynamic object) appears in its field of view. As long as the person moves within the field of view, all cells are updated correctly. But, when the person (or the robot) moves in such way that the previous position of the person is no longer in the field of view of the robot's sensor, the algorithm is unable to update the cell as free and thus the map is incorrect as it contains extra objects. Therefore, when planning the route, the algorithm plans a route that unnecessarily avoids obstacles that do not exist and the algorithm is slower.

The authors presented a way to deal with this problem. Besides the previously-defined *occupancy value* o_c they define also the *static occupancy value* s_c that represents the probability whether the cell is occupied *statically*. Unlike the o_c, s_c represents the long-term occupancy state and thus is updated

[7]The o_c is updated in a more complex way and we do not mention the rule here. It can be found in [4, p. 31]

more slowly than the o_c, however, the update formula [4, Eq. 13, p. 33] is similar.

The usage of the combination of o_c and s_c then allows to track moving objects (that are approximated by a normal distribution) and even to predict their movement. This results in solving the problem described above.

4.2.2 Graph-based SLAM

The mapping approach described above is then used in a graph-based SLAM. In the Section 3.4.3 we noted that during our research of literature we found two different approaches towards graph-based SLAM. The case when both robot positions and landmarks are represented by the vertices in the graph was described back there, now we are going to describe the case when only the robot positions are represented by the vertices.

The robot positions are represented as vertices of the graph. The sensors and odometry measurements represent constraints between two consecutive positions and are encoded by edges in the graph.

In this paper, the graph-based SLAM process is composed of two parts: *SLAM front-end* and *SLAM back-end*. These terms are used also by [17]. The SLAM front-end process is basically about building the graph and it can be further divided into 4 sub-processes: graph generation, loop-closure detection, map registration and fusion of vertices. All this is depicted in Figure 4.2.

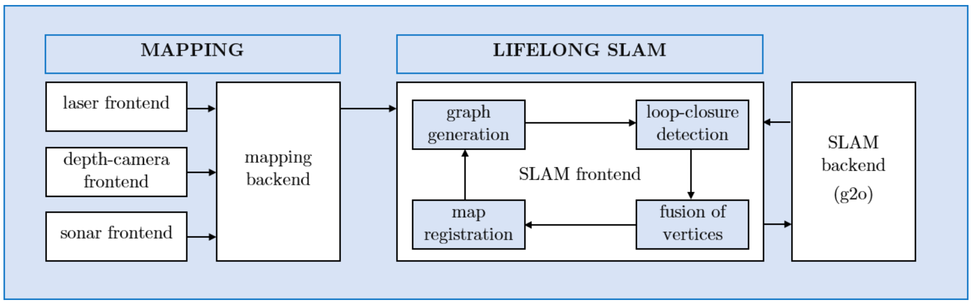


Figure 4.2: Process of mapping, drawn on the basis of [4, Fig. 7]

SLAM front-end

- **Graph Generation** Each vertex of the graph contains a small NDT map fragment of the whole map of the environment. The robot odometry is necessary in the mapping process. Odometry is not error-free, and the error gets bigger the longer distance is travelled. To deal with this problem, authors defined a specific threshold for the error. Once the threshold is exceeded, a new map fragment is started. As the new fragment is created, a new vertex is added to the graph and is connected with the previous vertex, having the odometry information encoded in the edge.

- **Loop-closure Detection** The work [24] defines loop-closure as "identifying a place observed previously but not very recently". The authors use a χ^2 test to determine whether two maps of consecutive vertices overlap significantly. If the test turns out positive, a special *loop-closure edge* is added into the graph. The optimization process (SLAM back-end) takes care of potentially incorrect loop-closures.

- **Map Registration** When merging two NDT maps, the goal is to minimize the distance between their normal distributions. The fact that the NDT maps are organized as 2^d-trees allows to perform the map registration not at the finest level of detail, therefore the process can be accelerated. Then, a second registration at the finest level can be executed.

- **Fusion of Vertices** For the lifelong usage of an algorithm, both the memory required and the computational time necessary must not grow. However, as the robot works continuously in the environment, new vertices with new map fragments are added and the size of the graph is increasing. This is why the fusion of vertices, whose map fragments represent the same part of the environment, is necessary.

  When two vertices are fused, the information stored in the newer one is transformed to the original one and all the neighbors of the newer one are connected to the original one. This ensures that the graph is reduced by at least one edge and one vertex, during each fusion. At the same time, the fact that we transform the information from the newer one to the older one makes sure that the most recent state of the environment will be represented in the graph.

Once a new vertex with a new map fragment is added, the process repeats.

SLAM back-end

The robot positions $x_1, x_2, \ldots, x_n$ are represented as vertices $v_1, v_2, \ldots, v_n$ in the graph. Two consecutive positions x_i, x_j are tied together in the real world by odometry measurements and sensor data. These limitations (often referred to as "constraints" in literature) are in the graph encoded in the specific edges of the graph. As the paper [4] says, "each constraint between two vertices v_i and v_j is represented by a transformation δ_{ji} that describes the pose x_j as seen from x_i and a corresponding information matrix Ω_{ji}". The SLAM problem is then handled by solving the following optimization problem:

$$X^* = \underset{X}{\operatorname{argmin}} \sum_{i,j} e(x_i, x_j, \delta_{ji})^T \; \Omega_{ji} \; e(x_i, x_j, \delta_{ji}) \tag{4.5}$$

where $e(x_i, x_j, \delta_{ji})$ represents the error function which tells us how well the guesses of positions x_i, x_j fit the constraint given by δ_{ji} (other symbols were explained above the equation).

The back-end of the SLAM process is responsible for the optimization of the graph that is given by Equation 4.5. There are several ways to tackle it. One of the most commonly used is the g2o[8], "an open-source framework for graph optimization" [4], as it was used not only by the authors of this state-of-the-art technique, but also by others. The output of this optimization process is the robot's trajectory of the highest likelihood.

4.2.3 Experiments

Several experiments using different sensors were conducted in [4]: a robot equipped with a Microsoft Kinect sensor in a home environment ($8 \times 5m$) and a robot with a laser range finder in an office building ($80 \times 35m$). All key aspects of good SLAM approach were taken into consideration when evaluating the results of the experiments:

- **Online** - Using a machine equipped with Intel Core i7 CPU (2.7 GHz), this approach is able to generate a new map every $0.5 - 1s$, which is sufficient for online usage.
- **Lifelong** - The ability of the algorithm to work lifelong was also tested by using guide robots equipped with a laser range finder only. The fact that a successful lifelong SLAM can be tackled using *only* the laser range finder is quite an interesting piece of information for our application. The lifelong-focused experiments were conducted for 2 days. The results show that the number of edges become more or less constant after some time.
- **Dynamics** - The results of the experiments show that this technique is able to deal both with highly dynamic objects and semi-static objects. According to the paper, it successfully detects persons moving in the environment without any inconsistencies in the map. The algorithm is also robust against changes in the environment, which was tested by moving furniture in the home environment while the robot was performing the mapping.

4.2.4 Conclusion

In general, the concept of NDT maps seems to be applicable. One of the benefits we see is that the amount of data does not grow - if we know the size of the whole environment and if we limit the size of the smallest cell, we get an upper boundary for the memory required.

4.3 Other Approaches

In this section we briefly look at three SLAM approaches we ran into during the research of literature that were somehow interesting, although not necessarily for their applicability to our challenge.

[8]https://openslam.org/g2o.html

4.3.1 Biologicaly Inspired SLAM

This SLAM approach is based on the idea that animals, for example mice, are able to localize and navigate in the environment throughout their lives - in other words they perform lifelong SLAM. And truly, their surroundings are dynamic - another animals live in it, weather and seasons change and so on. This idea resulted in presenting the *RatSLAM* algorithm which was used by [1] and others.

RatSLAM models part of rodent hippocampus[9]. The RatSLAM consists of three elements: pose cells, local view cells and experience map. Combined together, we can obtain the complete state (x, y, θ) of a robot within 3D environment.

These authors have presented their lifelong SLAM method at autonomous robot performing mock deliveries, which is in a way a similar application to ours. The results of the experiments performed were quite successful - the robot was active for 37 h totally, having travelled 40 km and autonomously recharged itself 23 times. In this time, it completed 1142 of 1143 delivery task with success. As the environment for the experiments, real-life setting of a busy office building was used. The robot had a laser sensor $(240°)$, 8 sonar sensors and also a camera $(360° \times 120°)$, which was facing in the direction of the robot's movement.

An interesting thought was brought up in this paper - whether the accuracy of a map is the key factor. Its Cartesian accuracy is important for SLAM, but the authors consider it secondary in the tasks of navigation from one place to another, which is the case of our industrial application as well. For UGVs traveling among several places in a factory, the map which is created during this process is not important. The key metric is whether the robot reaches its target destination or not. The experiments in this paper show that reliable navigation to various locations can be accomplished successfully without acquiring highly accurate maps.

SLAM as a Research Topic in Neuroscience

The above-mentioned approach is neither the first nor the last that takes the inspiration from biology. In fact, we may expect more nature-based techniques to appear in the future, as the professors in psychology and neuroscience M.-B. Moser, E. I. Moser & J. O'Keefe shared the Nobel Prize in Physiology or Medicine in 2014[10] for the discovery of the brain's positioning system. It is no surprise that an article by these authors was one of the references in [1].

[9]Hippocampus is a part of the brain which is located in the medial temporal lobe. Being part of the limbic system, it is important for spatial orientation and navigation and "transforming" information from the short-term to the long-term memory.

[10]The studies of spatial maps forming in brain go back to the first half of the 20th century.

4.3.2 Prediction-based SLAM

Chang [13] presented a novel approach towards the SLAM problem that examines the environment carefully and looks for a pattern in it. Based on this exploration it predicts what the environment is like in areas it has not explored yet, using patterns from already explored areas. Since some environments where mobile robots work may have a repetitive structure (such as office buildings, factories, shopping malls), the localization and mapping process can be successfully accelerated.

4.3.3 tinySLAM

Having less than 200 lines of code, this algorithm was designed by Steux & Hamzaoui [25] to be as minimalistic as possible, yet delivering satisfactory results. It uses laser scan data and odometry in a particle filter.

Chapter 5

Collective SLAM

As mentioned in Section 2.1, this thesis should help with the implementation of unmanned ground vehicles (UGV) in a factory for transporting objects between different places. The first level of improvement is to use more robots, so more objects can be moved and the logistics can be more efficient. However, even bigger improvements come when more robots cooperate, especially sharing information while performing a task. The more robots move through the factory, the more often the map of the environment is updated, thus it has higher accuracy. Furthermore, when for example an aisle is suddenly impossible to go through, the robot that senses it can inform the others. The other robots then avoid this and choose an obstacle-free path instead.

In the literature, such a cooperation is called *collective SLAM* (C-SLAM), *multi-robot SLAM*, *collaborative SLAM*, *multi-vehicle SLAM* and others. In this thesis we will mostly use the term C-SLAM.

A very brief description of C-SLAM is as follows: several mobile robots are moving in an environment, each building its *local map* of the environment and trying to localize within it. Once the robots know their relative poses, they can merge their local maps into one *global map*.

The advantages that are brought by using C-SLAM instead of SLAM include higher accuracy stemming from the redundancy of data. A collective robotic system is typically more robust to failures (as distributed system are in general). However these advantages on one side mean significantly increased complexity on the other side.

In this chapter we will look at the mathematical description of C-SLAM, the main challenges connected with it will be discussed and several approaches described, based on the research of literature. For deeper insights into this topic we can recommend work by Saeedi et al. [24] who have recently published an extensive and in-depth survey on the problem of multiple-robot SLAM.

5.1 Our Application Versus Literature

Our mobile robots (UGVs transporting objects between several places in the factory) will work there for a long period of time and, furthermore, will be equipped with an initial map of the environment. From our lifelong point of view, we see the main contribution of using multiple robots naturally in expanded transportation capacity, but particularly in increased ability to reflect the changes in the dynamic environment.

On the contrary, the majority of literature focusing on the C-SLAM topic sees its main advantages in faster exploration of an area. This difference between our application and the literature has to be reflected in the design of our future algorithms. During this chapter we sometimes refer the readers to this section, to keep these differences in mind.

5.2 Mathematical Formulation

To start simple, let us first denote a mathematical formulation of the C-SLAM problem for two robots only. Imagine that two robots are performing SLAM in an environment. Without loss of generality we can label the first robot $r^{(1)}$ and the second $r^{(2)}$.

As defined in the Equation 3.3, the SLAM problem for the robot $r^{(1)}$ is defined as follows:

$$p\left(x_t^{(1)}, m^{(1)} \;\middle|\; z_{1:t}^{(1)}, u_{1:t}^{(1)}, x_0^{(1)}\right) \tag{5.1}$$

and for the robot $r^{(2)}$ in a similar way:

$$p\left(x_t^{(2)}, m^{(2)} \;\middle|\; z_{1:t}^{(2)}, u_{1:t}^{(2)}, x_0^{(2)}\right) \tag{5.2}$$

where the symbols in the equation respect the notation from Section 3.1.1 with the extension of upper indexes $^{(1)}$ and $^{(2)}$ that assign the symbols to the first and second robot.

Now, let the robots work *cooperatively* on the SLAM problem - in other words let them work on the same map by putting

$$m^{(1)} = m^{(2)} = m \; . \tag{5.3}$$

Then, we can join the equations 5.1 and 5.2 as [24, Eg. 7, p. 4] did in order to obtain the equation for two-robot SLAM:

$$p\left(x_t^{(1)}, x_t^{(2)}, m \;\middle|\; z_{1:t}^{(1)}, z_{1:t}^{(2)}, u_{1:t}^{(1)}, u_{1:t}^{(2)}, x_0^{(1)}, x_0^{(2)}\right) \; . \tag{5.4}$$

Figure 5.1 depicts the two-robot SLAM situation. The same as in the Figure 3.1, squares represent observed variables and circles represent hidden ones.

Now we can move from the simple example of two robots to an example with n robots: $r^{(1)}, r^{(2)}, \ldots r^{(n)}$. It is obvious that the equation describing the SLAM problem for the robot $r^{(i)}$ will have the following form:

$$p\left(x_t^{(i)}, m^{(i)} \;\middle|\; z_{1:t}^{(i)}, u_{1:t}^{(i)}, x_0^{(i)}\right) \; . \tag{5.5}$$

Using this equation we can finally form the equation for the multi-robot SLAM problem:

$$p\left(\vec{x_t}, m \mid \vec{z}_{1:t}, \vec{u}_{1:t}, \vec{x_0}\right) \; , \tag{5.6}$$

where

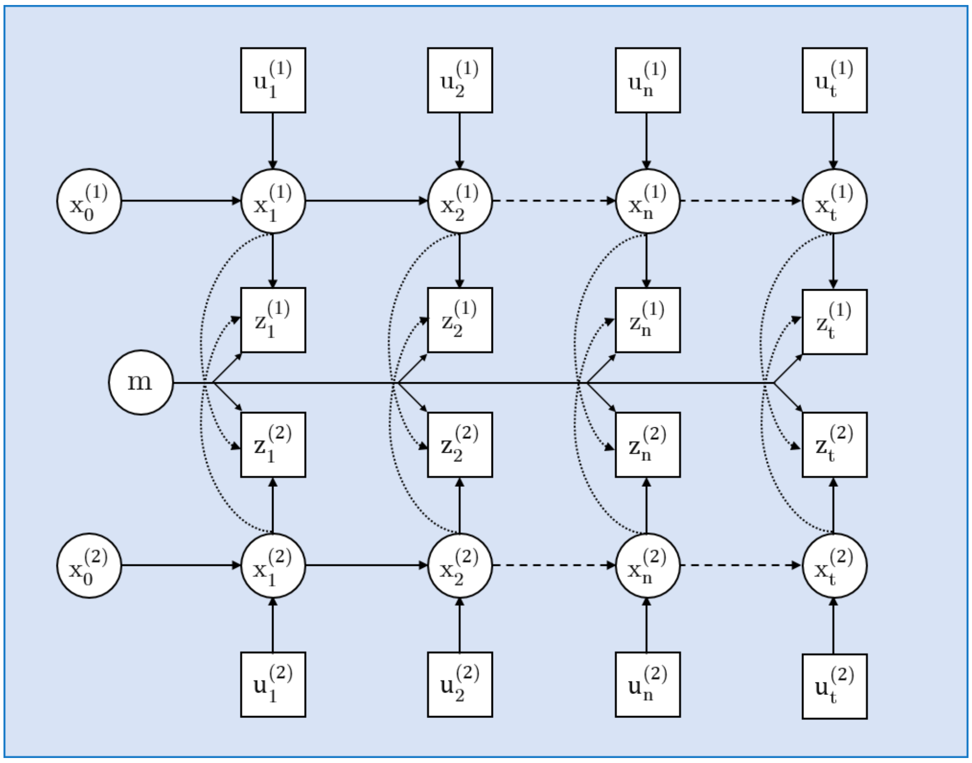


Figure 5.1: Two-robot SLAM process as dynamic Bayesian network, drawn on the basis of [24, Fig. 1, p. 5] and [20, Fig. 2, p. 2]

- $\vec{x_t} = (x_t^{(1)}, x_t^{(2)}, \ldots, x_t^{(n)})$ is the vector of all robots' positions at time t,
- m is the map of the environment,
- $\vec{z}_{1:t} = (z_{1:t}^{(1)}, z_{1:t}^{(2)}, \ldots, z_{1:t}^{(n)})$ is the vector of all robots' measurements,
- $\vec{u}_{1:t} = (u_{1:t}^{(1)}, u_{1:t}^{(2)}, \ldots, u_{1:t}^{(n)})$ is the vector of all robots' control inputs, and
- $\vec{x_0} = (x_0^{(1)}, x_0^{(2)}, \ldots, x_0^{(n)})$ is the vector of all robots' initial positions.

5.3 Main Challenges

The C-SLAM brings many advantages to the world of localization and mapping. However, a bigger number of robots also means increased complexity and many other challenges that do not have to be dealt with in the single-robot SLAM. Carlone et al. [21] listed 5 main challenges and Saeedi et al. [24] even 10. Below, some of them are discussed, both in general and in the context of our future industrial application[1].

The overall challenges arising from multiple-robot operation such as task allocation, formation and coordination of movement [24] and others have to be taken into account as well, however they are not described here, as they are not that relevant to the subject of this thesis.

[1]See Sections 2.1 and 5.1.

5.3.1 Initial and Relative Positions

There are two possible cases: either the initial positions of the robots are known, or they are unknown. The first scenario is much easier than the second one, however the latter is the more common one.

In the context of our application, we probably might know the initial positions of the robots at the beginning of the process. But we have to take into account the fact that all the robots will not always be in the working state - a robot might be moved away for maintenance or because of a failure. When returned back to the process of transporting objects, it will not know the positions of the others, thus we have to deal with the initial positions as unknowns.

5.3.2 Map Merging and Fusion

The problem of map merging is strongly connected with the challenge of initial and relative positions. Rone & Benz-Tvi [26] presented four ways of map merging, which differ in computation complexity and state knowledge, as the Figure 5.2 shows.

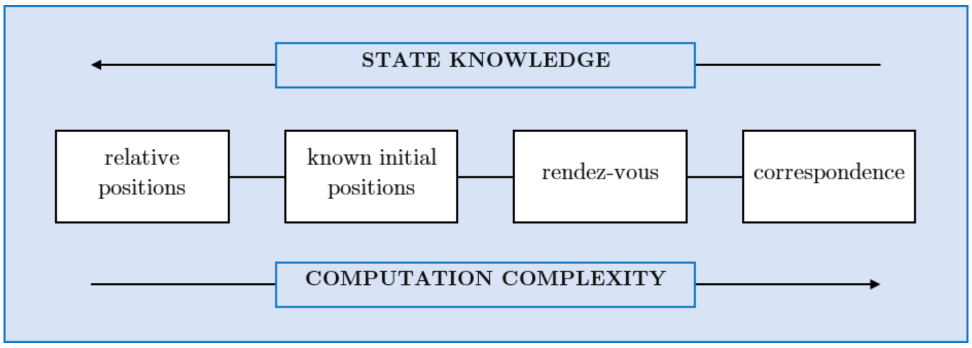


Figure 5.2: Various ways of map merging, drawn on the basis of [26, Fig. 10, p. 5]

If we know the *relative positions* of the robots with high accuracy, a relative transformation matrix, representing the transformation from the first robot's map to the second one's, can be constructed and consequently the global map can be built. Unlike the case of correspondence, we do not need the individual maps to overlap.

Having known the *initial positions* of the robot is in a way similar approach to map merging as the above-mentioned one. However the task of computing the robots' current positions based on their initial positions is not trivial.

The relative position of robots can also be computed when two robots "meet" (when one robot is detected within range of another robot's sensor). This meeting is referred to as a *rendezvous*. When a rendezvous happens, the robots are able to compute their relative positions necessary for the map merging.

If we have very low or zero knowledge of the relative positions of individual robots, we can use *correspondence* that merges overlapping maps based on the similarities in them. The drawback of this approach is the necessity of maps to overlap and the high computational complexity.

5.3.3 Communication Issues

A group of robots can cooperate only if the robots can communicate with each other, which is possible only when tools for reliable data sharing are available. The survey [24] stated four questions related to this challenge, which are answered in the following paragraphs.

1. What kind of data is shared?
2. How is the data shared?
3. Where is the data processed?
4. How is the data processed?

What Kind of Data Is Shared

There are two types of data that can be shared when performing a multi-robot task. The first way is to share the raw data (all sensor and odometry data). This approach requires the communication channel to be able to reliably carry large amounts of data. Also, we need more computational power to process all the data. When these conditions are met, we benefit from higher flexibility that the larger amount of data brings. An approach that shares the raw data was presented by Howard [20].

The other approach is to share data that is already processed - for example local maps. The amount of data to be shared is significantly smaller, as well as the computational complexity. One of the approaches that share processed data is for example work by Saeedi et al. [27].

Regarding our application, sharing of raw data does not make much sense, as the robots will be working in an environment with limited size - the communication channel would be overwhelmed by the amount of similar data. As a matter of fact, the information that the working area is limited should allow us to divide it into discrete cells[2]. Then the robots in the environment would only share information about cells whose state significantly differs from the state stored in a lifelong global map (shared among all robots) - let us call them *update data*. To use this approach, we would need to know the locations of all robots within the map with a very high accuracy, since even a small deviation would result in severe inconsistencies.

How Is the Data Shared

The channel for communication among robots has to be reliable, accessible by all robots and apparently wireless. An economic and reasonable solution is to use an existing communication infrastructure which the robots can easily

[2]This could be done for example by combining NDT maps and occupancy grid maps - the approach by [4] described in Section 4.2.1.

use and connect to. In our case that could be for example a Wi-Fi network in the factory. But as the Wi-Fi network is meant to be used mainly for other purposes than transferring data between robots, the amount of data transferred would have to be low. If the only data shared via the Wi-Fi were the *update data* mentioned in the previous paragraph, we assume that the Wi-Fi network could be used, without getting overwhelmed.

The case of an error in the communication channel should not be destructive to the whole C-SLAM process. Especially not in our application, where the shared information are not the cornerstone of the whole application, but rather an "icing on the cake" improving the whole process.

Where Is the Data Processed

Rone and Ben-Tzvi [26] introduced basic classification of multi-robot systems based on their architecture: they can be centralized, hierarchical or decentralized. The centralized approach requires one specific robot or another subject to be responsible for the task. The hierarchical architecture is inspired by the arrangement of military units [16]. In the decentralized approach the computation of the task is distributed among more robots. If we combine parts of these three architectures, we will create a hybrid architecture.

Because we expect the robots to be moved away for maintenance, for our application we would probably choose the centralized architecture where the responsible unit is not a robot, but a computer located somewhere in the factory. This unit would be responsible for carrying the global map of the environment. Although the responsible unit would be dependent on the mobile robots, we do not want this dependency to be mutual, as an error in the unit would collapse the whole system, which is undesirable.

How Is the Data Processed

Once all the data is successfully transferred among the robots and ready to be processed, the challenge of *how* to process it appears. A very brief introduction to C-SLAM approaches in general is located in Section 5.4.

5.3.4 Dynamics of the Environment

The algorithms for C-SLAM should be able to deal with dynamics in the environment. It is quite obvious, since any environment where more than one robot performs an activity is a dynamic one, because it contains the other robots - dynamic objects. The C-SLAM algorithms are typically based on one-robot SLAM algorithms, thus using an algorithm suitable for dealing with a dynamic environment as the base should be a sufficient condition to ensure stability in dynamic environments.

5.3.5 Computational Complexity

The redundancy of data is the main reason of increased computational complexity of C-SLAM algorithms in comparison with SLAM algorithms. The complexity is strongly dependent on the number of robots and the size of the map has an effect as well. Thus, when processing data from several robots, an efficient way to do so has to be found in order to make it possible for the algorithm to work online.

5.4 Approaches

The C-SLAM approaches and algorithms are based on the SLAM algorithms for one robot [24], but they have to deal with new challenges stemming from the presence of multiple robots which were described in the previous section.

The same as in the case of single-robot SLAM, some authors use EKF or other filters, such as Sparse Extended Information Filter used by Thurn & Liu [28].

Also the particle filters are used very often, especially RBPF[3], despite the fact that they are quite challenging because of the amount of data carried - each robot has one map hypothesis for one particle. Some of the C-SLAM approaches based on the RBPF [21] follow Grisetti, Stachniss & Burgard [2], whose work we have chosen as the state-of-the-art approach and described it in Section 4.1. Other works that follow a single-robot RBPF approach include for example [20], [22] or [13].

[3]Rao-Blackwellized Particle Filters, for description see Section 4.1.1.

Chapter 6

Experimental Setup and Experiments

In this section we describe both the background (experimental setup) and the process (experiment) of measuring real-life data that will be used for the design and testing of a lifelong SLAM algorithm. We explain what software and hardware we used and why, and we also suggest a specific Robot Operating System (ROS) packages that can be used.

6.1 Hardware

Regarding our future industrial application, the robot and the sensors are the hardware elements that have to be taken into consideration. In this thesis, we focus on the LIDAR.

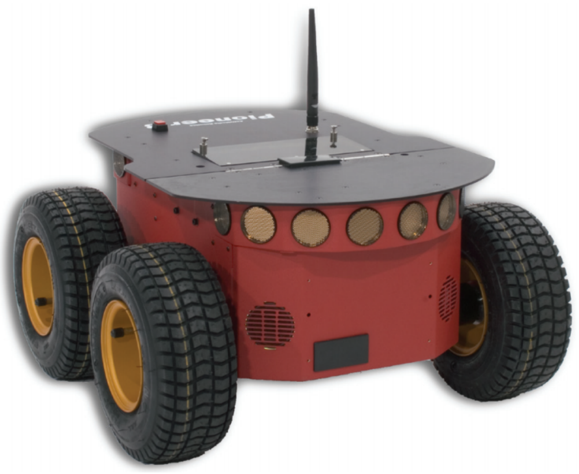


Figure 6.1: Mobile robot Pioneer 3-AT, figure from [29]

6.1.1 Pioneer 3-AT

So far we only have a rough idea of what kind of wheeled mobile robots (UGVs) will be used in our industrial application in the future. But for the purpose of designing, implementing and testing the lifelong algorithm, basically any wheeled mobile robot can be used. For our experiments we have chosen Pioneer 3-AT, as it was available from the CIIRC. The robot is in the

Figure 6.1. The Pioneer family of robots is used by many researchers, such as [2], [30], [31], [5], [1], [32], which means that variety of reliable software tools are available, including the Robot Operating System (ROS), which is the subject of Section 6.2.

The Pioneer 3-AT is a four-wheeled mobile robot produced and delivered by Adept Mobilerobots company[1]. Its steering is achieved by applying different velocities for each side pair of the wheels - so called *skid-steer locomotion*. This type of movement negatively affects the odometry.

According to its datasheet [29] it should be able to go as fast as $0.8\,\mathrm{m\,s^{-1}}$ and should be able to run on battery power for up to 4 h. It has a built-in security feature - rear and front bumpers located slightly above the floor that automatically stop the motion of the robot when they hit an obstacle.

6.1.2 SICK LMS111

Unlike the mobile robot, the choice of sensors to use is crucial. It determines the characteristics of the data used as inputs for the algorithms of localization, navigation and mapping. We have already discussed the sensors in general in the Section 2.2 and we introduced some arguments why we have chosen the laser range finder SICK LMS111 in the Section 2.2.2. Now we are going to look at the sensor in detail.

Figure 6.2: Laser range finder SICK LMS111, figure from [33]

[1]http://www.mobilerobots.com/Mobile_Robots.aspx

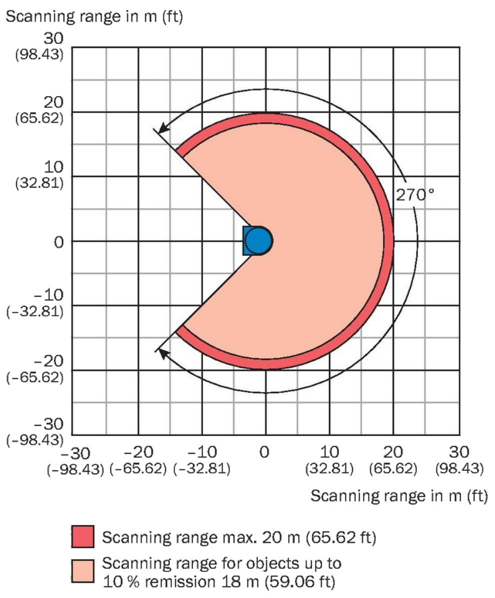


Figure 6.3: LMS 111: operating range diagram, figure from [33]

LMS111 is a high-performance highly robust sensor. In our experiments we used the type LMS111-10100, which is designed to work also in an outdoor environment. However, in the future application the indoor variant of this sensor (LMS111-10000) will be used. Regardless of the ability to work outdoors (ensured by functions such as heating and wider interval of permissible operating temperature), all the other key parameters are equal for the outdoor and indoor versions of the LMS111 sensor.

The basic technical details of LMS111-10100 are shown in the Table 6.1, more to be found in [33]. For the diagram of the operating range, see Figure 6.3. Regarding safety, the device is part of the laser class 1, thus it is eye-safe.

Aperture angle	270°
Operating range	0.5 m to 20 m, maximum 50 m
Angular resolution	0.25° / 0.5°
Scanning frequency	25 Hz / 50 Hz
Errors of measurement	systematic error ± 30 mm with ± 15 mm of statistical error
Interfaces	Serial (RS-232) - 9.6 kBaud to 115.2 kBaud Ethernet - 10/100 Mbit/s CAN bus
Operating voltage	10.8 V to 30 V DC
Power consumption	8 W
Weight and dimensions	1.1 kg, 105 × 102 × 162 mm

Table 6.1: The technical specification of LMS111-10100

Operating Principle

The LMS111 is a 2D laser measurement sensor[2] that scans its surroundings. Using a laser diode and a rotating mirror, it emits infrared laser pulses with a wavelength of 905 nm. It measures reflected laser pulses within the angular range of 270°.

The scanning distance range is dependent on the object's reflectivity - we can detect objects that are as far as 50 m, but only in optimal conditions. In order to reliably detect an object with a reflectivity of 10 %, the distance between the object and the sensor cannot be bigger than 18 m.

Once the pulse hits an object, it is reflected back to the sensor, where it is detected by a photodiode. Then, the distance of the object is measured based on the time of flight and according to Equation 2.1. The operating principle is shown in the Figure 6.4.

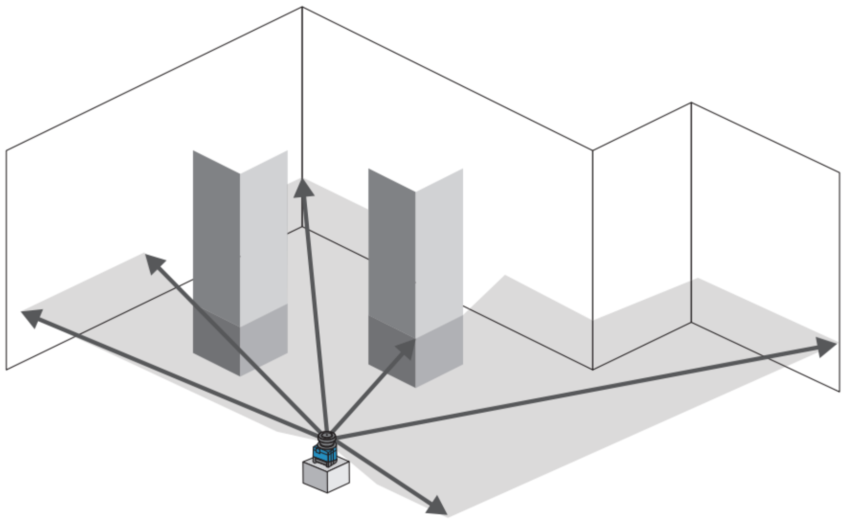


Figure 6.4: LMS111: operating principle, figure from [33, p. 23]

Beam Diameter and Minimum Object Size

The size of the laser beam is dependent on the distance from the sensor, as the Figure 6.5 shows. This affects the minimum object size that can be detected. For a successful detection of an object, the object has to be bigger than the diameter of the laser beam - the beam has to be incident on it totally. If this condition is not met, the object cannot reflect the whole energy of the laser and consequently the measurement can be lost.

The diameter of the beam d [mm] in the distance l [mm] is:

$$d = 8 \text{ mm} + l \times 0.015 \text{ rad} \ . \tag{6.1}$$

[2]This is where the abbreviation LMS in its name comes from.

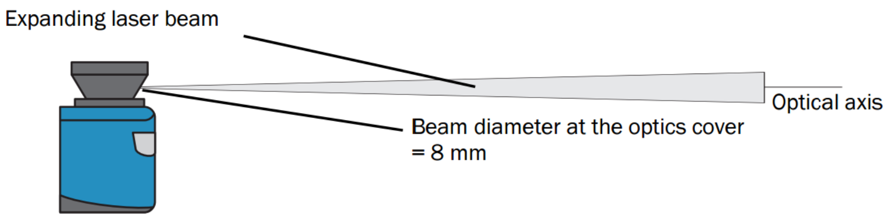


Figure 6.5: LMS111: beam diameter, figure from [33, p. 28]

The distance between two consecutive measurements k [mm] is also distance-dependent and further dependent on the angular resolution $\Delta\omega$ [rad]:

$$k = l \times \Delta\omega \ . \tag{6.2}$$

Using Equations 6.1 and 6.2 we get the formula for the minimum object size s_{min} [mm] that can be reliably detected at the distance l [mm] with the angular resolution $\Delta\omega$ [rad]:

$$s_{min}(l, \Delta\omega) = d + k = 8 \text{ mm} + l \times (0.015 \text{ rad} + \Delta\omega) \ . \tag{6.3}$$

Communication Interface

We communicate with the sensor through the Robot Operating System via the Ethernet interface. However, there are two more ways how to send telegrams via terminal program to the LMS100 sensor - ASCII or Binary. Both ways are more or less equivalent and the sensors answers in the same "language" which was used to talk to it. It is all covered in detail in [34].

Overall, these kind of sensors are often used for the SLAM challenge, for example in [2], [3], [4]. Although alternative sensors exist and at the same time the SICK sensors are expensive, we have decided to choose these anyway because of their robustness, reliability, deep documentation and available support.

6.2 Software

Robot Operating System (ROS) is an open-source software framework for robotics. Not only does it provide engineers and programmers with packages, tools, drivers, libraries and other features that make the development of a robot application easier and can save many developer hours, there is also a big community around it - conferences (ROSCon), forums, tutorials and others.

The ROS has been around since 2007 [35], but its popularity has started to grow just in the last few years, maybe due to its rapid development - a new distribution was sometimes released even twice a year which made its usage complicated. The pace of releasing new versions is getting slower[3], however, during the months we worked on this thesis, we were considering three different distributions. In the beginning, *Indigo* was suggested by the authors of the ROS framework as the most stable way, later *Jade* became the recommended distribution and recently a new distribution *Kinetic* was released. Eventually, we have chosen *Jade* for our work and used it on a laptop with Ubuntu 14.04.

There are many other frameworks for robotics software, Kramer & Scheutz [37] compared 11 of them back in 2007. However, ROS is probably the most popular nowadays as it "probably has hundreds of thousands of active users" [38]. Nevertheless, some developers still prefer to build their applications from scratch, so it fits their needs perfectly.

6.2.1 How ROS Works

The ROS is well-structured so it could provide its users and contributors with the desired level of abstraction. The whole structure is outlined in the Figure 6.6, its basic elements are: packages, nodes, topics, messages and services.

- **Packages** are units of ROS software that represent a useful piece of software[4]. Packages consist of nodes, usually of many nodes.
- **Nodes** are executables that perform computation [39]. For the communication between nodes messages are used, which can be transferred using either topics or services.
  - **Master** makes it possible for nodes to "see" each other and it also provides the name registration [40].
  - **Parameter server** is a part of the master and basically stores all the data.
- **Messages** are just structured information data that travels between nodes. There are different types of messages for different data. A type of message can be composed of other message types.
- **Topics** are elements that nodes can publish to or subscribe to. More than one node can publish to a topic and also subscribe to a topic. The topics and the publisher/subscriber architecture are an elegant way to share data among nodes.
- **Services** represent another way of how nodes can get information. Services are provided by nodes. When a node wants to "use" the service, it has to send a specific request message to it, and then it receives a reply message.

[3]As of 25. May 2016, the ROS community plans to release a new version in May of each year. [36]

[4]Currently over 1,000 packages for the *Jade* distribution is available to be downloaded from http://www.ros.org/browse/list.php

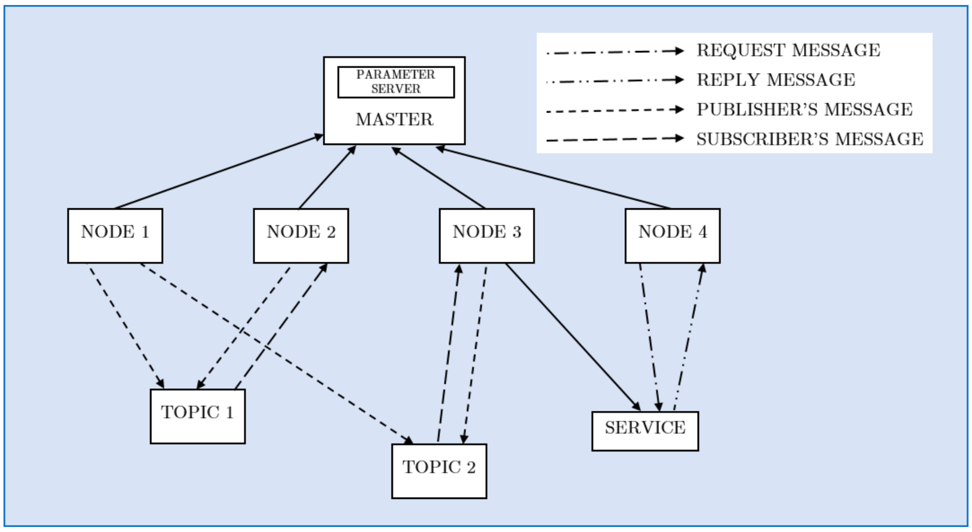


Figure 6.6: The communication within the ROS structure

6.2.2 Used Packages

The fact that ROS is quite popular and thus has a lot of developers and users is double-edged, as there are always several ways (sometimes even too many ways) how to reach a single goal. In the following paragraphs we present and recommend the packages we used with success for our experiments. We also share the ROS commands that we used in order to make the potential reimplementation easier.

ROSARIA

The *ROSARIA* package [41] is one of the essential packages as it connects the ROS with the Pioneer 3-AT and other Adept MobileRobots. It enables access to all the important data from the mobile robot, such as odometry information that is being published to the */RosAria/pose* topic. It is subscribed to the */cmd_vel* topic, therefore it receives velocity commands (via this topic) that control the motion of the robot. The documentation also describes a way to connect the laser and get its data, however, after some struggling we decided to use another package for that task.

We use the following command to run the ROSARIA node.

```
rosrun rosaria RosAria
```

Once the RosAria node is running, we can read the odometry information easily.

```
rostopic echo /RosAria/pose
```

Teleop_twist_keyboard

The ROSARIA package also provides a way to tele-operate the robot which is however not very useful, as only 4 types of movement commands can be used (forwards, backwards, spin right, spin left). This is why we have chosen the *teleop_twist_keyboard* package[42] for the control of the robot's movement. The motion is controlled using only 9 keys, which makes the control very simple, yet sufficient for our experiments.

The following command is used to run this node.

```
rosrun teleop_twist_keyboard teleop_twist_keyboard.py
```

LMS1xx

This node [43] allows us to connect the ROS with a SICK LMS 1xx laser and to get (read) its laser data that is published to the */scan* topic.

Firstly, we have to define the IP address of the Ethernet port of the computer running ROS to be the same as the IP address of the sensor[5]. Then, we can connect to the laser. Once the *LMS1xxx* node is running, we can read the laser-scan data easily.

```
sudo ifconfig eth0 10.10.20.1 netmask 255.255.255.0
rosrun lms1xx LMS1xx_node _host:=10.10.20.181
rostopic echo /scan
```

Rviz

The `rostopic echo` command is not a very elegant way to look at data, as it just causes it to print to the terminal. It is much better to visualize the data, which we did using the rviz package [44]. The package provides advanced 3D visualizations for anything that happens in the ROS topics. After the package is launched, which is done easily with a simple command `rviz`, we just define the topics we want to display. An example of the visualization output can be found in the Figure 6.11.

Rosbag

The rosbag package [45] enables recording any data that is published to any topic and then also to play back this data. This comes in handy particularly when designing and algorithm. The user measures real-life data just once and then he can play it back in order to test the algorithm, so he does not have to physically run the robot every time he wants to try the algorithm.

To record the odometry data from the */RosAria/pose* topic and the laser scan data from the */scan* topic and to save them into file *recording_test* we

[5]Or vice versa - to define the IP address of the sensor to be the same as the IP address of the Ethernet port, this can be done using the Configuration & Diagnostic Software provided by SICK.

used the following command.

```
rosbag record -O recording_test /RosAria/pose /scan
```

6.3 Experimental Setup

Having explained the hardware and software used for our experiment, we can proceed further to the description of the experimental setup. The overall structure of the setting of both hardware and software can be found in the Figure 6.7. The photo of the robot we actually used is depicted in the Figure 6.8. Since the robot is also used by other research groups, there is a large number of sensors mounted on it. As we mentioned several times, we were using data from the LIDAR only.

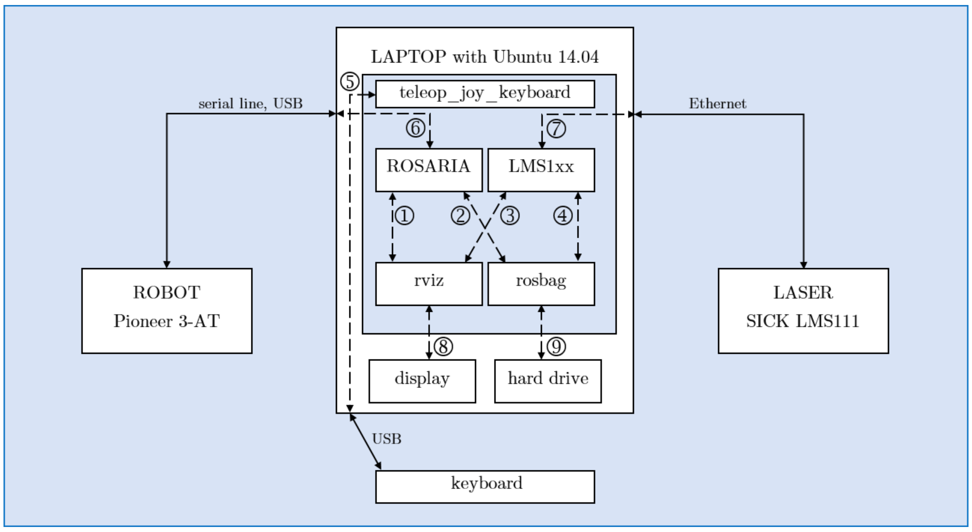


Figure 6.7: The setting of our experiment: the blue rectangle in the LAPTOP rectangle stands for the ROS, the lines ① to ④ represent ROS topics and messages transferred via them: ①, ② represent the odometry information in the topic */RosAria/pose*, ③, ④ represent the laser scan data in the topic */scan*, and finally the lines ⑤ to ⑨ represent the ROS packages that connect the hardware and the data.

6.4 Experiments

The goal of the experiments we performed was to evaluate the hardware and software that could be used either for the future industrial application or for its development. We have already described one experiment in Section 2.2.3, which we conducted in order to explore the possibilities of using the laser sensor SICK S300 for both safety and measurement purposes.

In this section we describe the experiments conducted using mobile robot Pioneer 3-AT, laser range finger SICK LMS111 and the Robot Operating System on the software side. The objective was to gather odometry and laser data from real-world environment which could be used for further research.

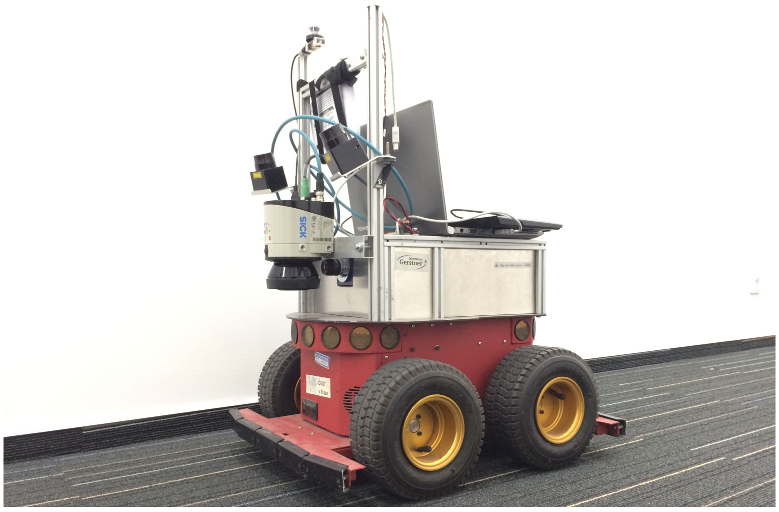


Figure 6.8: Photo of our experimental setting

6.4.1 Measured Data Format

Odometry Data

The odometry information is in the ROS published at the *RosAria/pose* topic in *nav_msgs/Odometry* file format. Its structure is depicted in the Figure 6.9. The rate at which the odometry information is published is 10 Hz [41].

Laser Data

The data from the laser range finder SICK LMS111 are publied at the */scan* topic in ROS in *sensor_msgs/LaserScan* format. The structure of this format is shown in the Figure 6.10. The frequency of publishing is set when configuring the laser scan, in our case it is 5 Hz. The example of data published from the hall in the BLOX office building is depicted in the Figure 6.11.

```
nav_msgs/Odometry
├─std_msgs/Header .....................header information
│ ├─uint32 seq .......................ID (auto incremented)
│ ├─time stamp .......................time
│ │ ├─int secs .......................Unix epoch time [s]
│ │ └─int nsecs .....................time since "secs" [ns]
│ └─string frame_id .................frame for the data
├─string child_frame_id ..............child frame id
├─geometry_msgs/PoseWithCovariance ..pose in free space
│                                     with uncertainties
│ ├─geometry_msgs/Pose ..............pose in free space
│ │ ├─geometry_msgs/Point ..........Cartesian position
│ │ │ └─float64 x, y, z
│ │ └─geometry_msgs/Quaternion .....quaternion orientation
│ │   └─float64 x, y, z, w
│ └─float64[36] .....................6x6 covariance matrix
└─geometry_msgs/TwistWithCovariance .velocity in free space
                                      with uncertainties
  ├─geometry_msgs/Twist..............velocity in free space
  │ ├─geometry_msgs/Vector3.........linear velocity
  │ │ └─float64 x, y, z
  │ └─geometry_msgs/Vector3.........angular velocity
  │   └─float64 x, y, z
  └─float64[36] .....................6x6 covariance matrix
```

Figure 6.9: Structure of odometry data, based on [46]

```
sensor_msgs/LaserScan
├─std_msgs/Header ............header information
│ ├─uint32 seq ...............ID (auto incremented)
│ └─time stamp ...............time
│   ├─int secs ..............Unix epoch time [s]
│   └─int nsecs .............time since "secs" [ns]
├─float32 angle_min ..........start angle of scanning [rad]
├─float32 angle_max ..........end angle of scanning [rad]
├─float32 angle_increment ....distance between consecutive
│                              measurements [rad]
├─float32 time_increment .....time between consecutive
│                              measurements [s]
├─float32 scan_time ..........time between consecutive
│                              scans [s]
├─float32 range_min ..........minimal range [m]
├─float32 range_max ..........maximal range [m]
├─float32[] ranges ...........range of measured data [m]
└─float32[] intensities ......intensities of measured data
```

Figure 6.10: Structure of laser data, based on [47]

6.4.2 Measuring Real-Life Data in Office Building

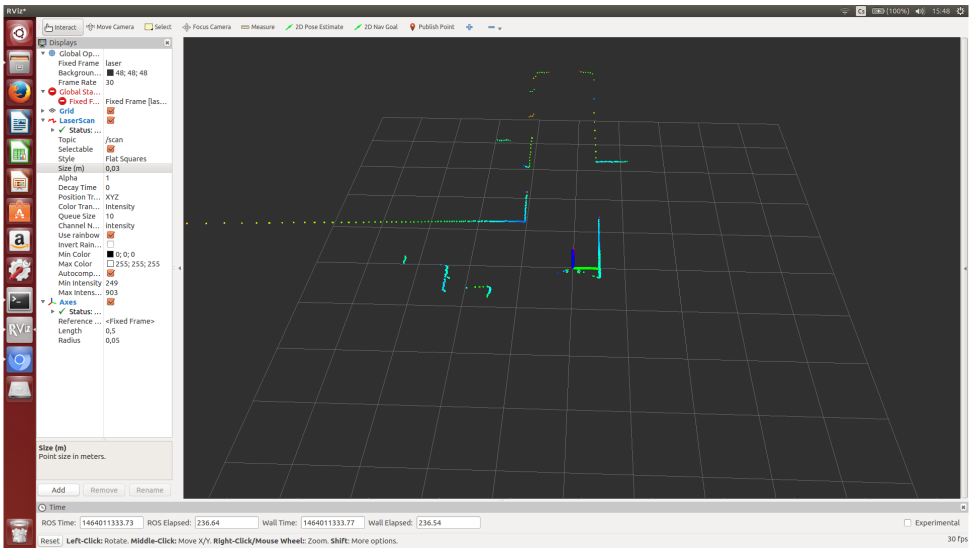
The measurements were taken in the office building BLOX in Prague 6, where the CIIRC is currently situated. Despite that corridors in office buildings are not very dynamic, we have tested the motion control of the robot and the behavior of the laser - how it registers its surroundings with some basic changes in the environment, such as people walking by or opening and closing of doors.

We conducted several trial rides in the corridors and eventually recorded two of them for a future use. The first one was a 96 s loop ride in quite a uniform environment. The second one was 270 s long and it was again looped. During the second run the laser observed opening and closing doors, walking persons, walls made of glass and the whole surrounding of the path was bit more irregular. The print screen of the vizualization of the laser "view" in the *rviz* and the place, from which it was taken, are depicted in the Figures 6.11 and 6.12.

The gather data are saved in `bag`[6] file format (`BLOX_1.bag`, `BLOX_2.bag`) and are part of the enclosed CD[7]. The Figures 6.13 and 6.14 shows examples of one message being published on topic */RosAria/pose* and */scan*, respectively. The general structure of this data was described in the previous section.

[6]`bag` files are produced by the *rosbag* package described in Section 6.2.2.

[7]See Appendix A.



Global Op...	
Fixed Frame	laser
Backgroun...	48; 48; 48
Frame Rate	30
Global Sta...	
Fixed F...	Fixed Frame [las...
Grid	✓
LaserScan	✓
Status: ...	
Topic	/scan
Selectable	✓
Style	Flat Squares
Size (m)	0,03
Alpha	1
Decay Time	0
Position Tr...	XYZ
Color Tran...	Intensity
Queue Size	10
Channel N...	intensity
Use rainbow	✓
Invert Rain...	☐
Min Color	0; 0; 0
Max Color	255; 255; 255
Autocomp...	✓
Min Intensity	249
Max Intens...	903
Axes	✓
Status: ...	
Reference ...	<Fixed Frame>
Length	0,5
Radius	0,05

Size (m)
Point size in meters.

ROS Time: 1464011333.73 ROS Elapsed: 236.64 Wall Time: 1464011333.77 Wall Elapsed: 236.54 ☐ Experimental

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click/Mouse Wheel:**: Zoom. **Shift:** More options. 30 fps

Figure 6.11: Print screen of visualization of laser data in *rviz*

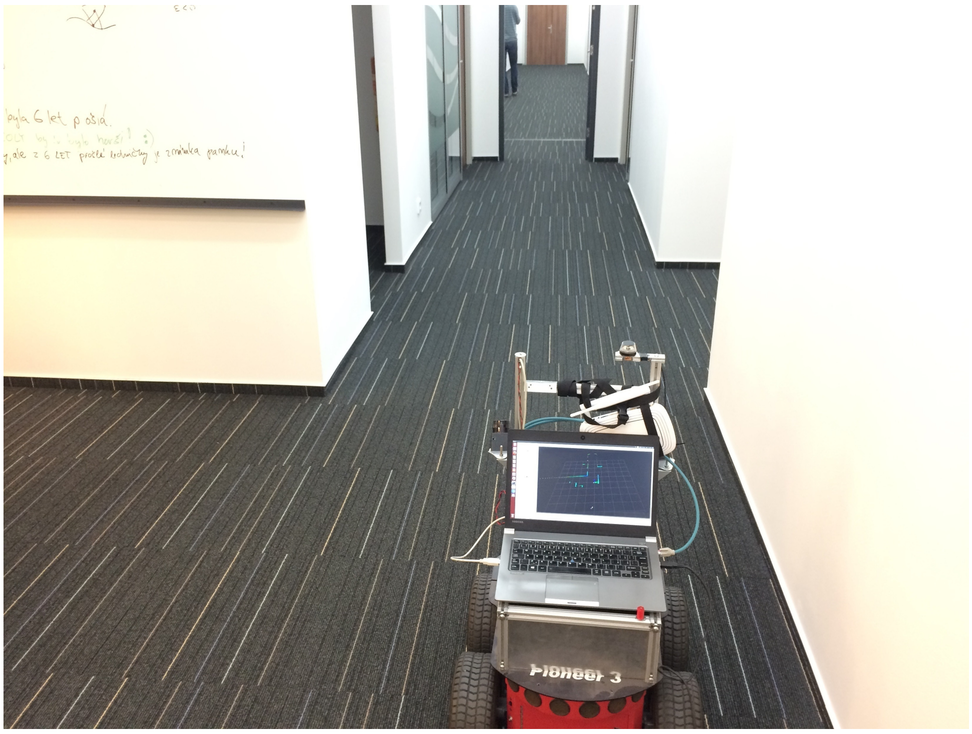


Figure 6.12: The position in which the Figure 6.11 was taken

```
header:
   seq:  3539
   stamp:
      secs:  1464011405
      nsecs:  362150682
   frame_id:  odom
child_frame_id:  base_link
pose:
   pose:
      position:
         x:  4.26312
         y:  -9.1791
         z:  0.0
      orientation:
         x:  0.0
         y:  0.0
         z:  0.932419047753
         w:  0.361378913867
   covariance:  [0.0, 0.0, ..., 0.0, 0.0]
twist:
   twist:
      linear:
         x:  0.7015
         y:  0.0
         z:  0.0
      angular:
         x:  0.0
         y:  0.0
         z:  0.0051
   covariance:  [0.0, 0.0, ..., 0.0, 0.0]
```

Figure 6.13: Example of odometry data published at */RosAria/pose* topic

```
header:
   seq:  3539
   stamp:
      secs:  1464011405
      nsecs:  362150682
   frame_id:  odom
angle_min:  -2.35619449615
angle_max:  2.35619449615
angle_increment:  0.00872664619237
time_increment:  2.77777780866e-05
scan_time:  0.019999999553
range_min:  0.00999999977648
range_max:  20.0
ranges:  [0.15299999713897705, 0.14800000190734863, ...,
          0.15199999511241913, 0.16099999845027924]
intensities:  [515.0, 532.0, ..., 526.0, 511.0]
```

Figure 6.14: Example of laser data published at */scan* topic

6.5 SLAM Algorithms Available in ROS

As the simultaneous localization and mapping is a popular research topic in the world of mobile robotics, it is no surprise that the ROS offers packages that deal with this challenge. Over time the following packages appeared: *gmapping*, *hector_slam*, *LagoSLAM*, *coreslam*, *tinySLAM*, *karto*, *vslam*, *rgbdslam*, *stereo_slam* and others.

Back in 2013, Santos et al. [14] compared 5 SLAM algorithms[8] available in ROS[9] in both simulations and real-world environments using an Arduino Based robot and laser range finder. The created maps were compared with the actual environment using the k-nearest neighbor approach. The lifelong aspect was taken into consideration, as the CPU load was measured and evaluated. All five algorithms produced consistent maps in simulations as well as in the real-world settings.

We mentioned that the ROS is developing fast and new distributions have been appearing up to twice a year. Therefore it is without big surprise that for the distribution of ROS we used (*Jade*), only three SLAM packages are available: *gmapping*, *hector_slam* and *stereo_slam*. Though the future industrial application will probably need a custom-made lifelong SLAM algorithm, we might use these open-source algorithms for initial experiments, therefore we provide a brief description of them.

6.5.1 Gmapping

According to [14], *gmapping* [48] is the most used SLAM package available in ROS worldwide. It is an open-source implementation of [2], which was described in the Section 4.1.

6.5.2 Hector_slam

Hector_slam [49] estimates the pose based on scan matching. It is able to work online and does not use the odometry information. In the tests conducted by [14], *hector_slam* together with *gmapping* had the best results. Both algorithms have also had reasonably low and comparable CPU requirements.

6.5.3 Stereo_slam

Stereo_slam [50] uses only one stereo camera, thus it is not very interesting for our approach. It addresses the SLAM problem with the graph-optimization approach.

[8]hector_slam, gmapping, karto, coreslam and LagoSLAM

[9]At that time, *Fuerte* was the distribution used.

Chapter 7

Conclusion

Throughout this thesis we were examining the topic of lifelong localization of mobile robot in the context of a future implementation of UGVs for transporting materials in a factory.

In the Chapter 2 we described key aspects of UGVs in industry with focus on sensors. The reasons for choosing laser range finder as the main sensor for out application were explained. We also conducted an experiment with safety laser scanner SICK S300 whose results show that the sensor is able to fulfil both safety and measurement purposes. Therefore we do not have to equip the UGVs with two costly sensors.

The Chapter 3 focused on the formal description of lifelong localization and related topics. The Chapter 4 was devoted to the state-of-the-art algorithms suitable for solving our problem. Works [2] and [4] were both described in detail. The first approach that builds on RBPF is used widely by the mobile robotics community and is available online for download. One of the assets of the second work lies in their hybrid NDT mapping approach, which gives an upper boundary for the memory required, which is one of the key factors of lifelong localization.

Collective SLAM, the topic of Chapter 5, deals with cooperation of more mobile robots on a task. The main issues that have to be considered were discussed, again with accent on our future application, since the differences between the literature and our challenge are in the case of C-SLAM quite significant.

Finally, the Chapter 6 describes the experiments we conducted. Used hardware and software is described closely and its usability for the future application is evaluated. The ROS strengthens its position in the mobile robotics community, we suggest to use it at least for further experiments and testing of algorithms. On the hardware side, the laser range finder SICK LMS111 seems to be applicable. In order to collect real-world data for further research, we conducted experiments in an office building, having used mobile robot Pioneer 3-AT (it will not be used in the factory, though). The collected data are described briefly and can be found on the CD.

Although we digressed a bit from the initial guidelines by putting the main emphasis on the research of literature, we believe this thesis is a contribution, as it should save hours of research of the implementation team.

Bibliography

[1] G. W. Michael Milford, “Persistent navigation and mapping using a biologically inspired SLAM system,” *The international journal of robotics research*, vol. 29, pp. 1131–1153, Aug. 2010.

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

[3] G. D. Tipaldi, D. Meyer-Delius, and W. Burgard, “Lifelong localization in changing environments,” *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1662–1678, 2013.

[4] E. Einhorn and H.-M. Gross, “Generic NDT mapping in dynamic environments and its application for lifelong SLAM,” *Robot. Auton. Syst.*, vol. 69, pp. 28–39, Jul 2015.

[5] S. Hochdorfer, M. Lutz, and C. Schlegel, “Lifelong localization of a mobile service-robot in everyday indoor environments using omnidirectional vision,” pp. 161–166, 2009.

[6] “SICK S300 Safety laser scanner - Operating Instructions.” `https://www.sick.com/media/dox/3/13/613/Operating_instructions_S300_Safety_laser_scanner_en_IM0017613.PDF`, Oct 2013.

[7] “SICK S300 Safety laser scanner - Telegram Listing.” `https://www.sick.com/media/dox/2/92/892/Telegram_listing_S3000_Standard_Advanced_Professional_Remote_S300_Standard_Advanced_Professional_de_en_IM0022892.PDF`, Feb 2015.

[8] Český normalizační institut, “ČSN EN 1525 Bezpečnost motorových vozíků - vozíky bez řidiče a jejich systémy,” jun 1998.

[9] B. Siciliano and O. Khatib, eds., *Springer Handbook of Robotics*. Springer, 2008.

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

[11] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[12] C. S. W. B. G. Grisetti, R. Kümmerle, "A Tutorial on Graph-Based SLAM," 2010.

[13] H.-J. Chang, "Simultaneous localization and mapping algorithms with environmental-structure prediction for single- and multi-robot systems," 2007. Copyright - Copyright ProQuest, UMI Dissertations Publishing 2007; Poslední aktualizace - 2015-08-26; První strana - n/a.

[14] J. Machado Santos, D. Portugal, and R. P. Rocha, "An evaluation of 2D SLAM techniques available in Robot Operating System," pp. 1–6, IEEE, 2013.

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

[16] M. K. Petr Štěpán, "Materiály k předmětu A3M33MKR - Mobilní a kolektivní robotika [online]." `https://cw.fel.cvut.cz/wiki/courses/A3M33MKR/start`, 2014. Retrieved: 2016-05-23.

[17] H. Kretzschmar, G. Grisetti, and C. Stachniss, "Lifelong map learning for graph-based SLAM in static environments," *KI - Künstliche Intelligenz*, vol. 24, no. 3, pp. 199–206, 2010.

[18] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[19] T. Werner, *Optimalizace - elektronická skripta předmětu A4B33OPT*. České vysoké učení technické, 2016.

[20] A. Howard, "Multi-robot simultaneous localization and mapping using particle filters," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1243–1256, 2006.

[21] L. Carlone, M. Kaouk Ng, J. Du, B. Bona, and M. Indri, "Simultaneous localization and mapping using rao-blackwellized particle filters in multi robot systems," *Journal of Intelligent & Robotic Systems*, vol. 63, no. 2, pp. 283–307, 2011.

[22] A. Gil, Óscar Reinoso, M. Ballesta, and M. Juliá, "Multi-robot visual {SLAM} using a rao-blackwellized particle filter," *Robotics and Autonomous Systems*, vol. 58, no. 1, pp. 68 – 80, 2010.

[23] H. Moravec, "Sensor fusion in certainty grids for mobile robots," *AI Mag.*, vol. 9, pp. 61–74, July 1988.

[24] S. Saeedi, M. Trentini, M. Seto, and H. Li, "Multiple-robot simultaneous localization and mapping: A review," *Journal of Field Robotics*, vol. 33, no. 1, pp. 3–46, 2016.

[25] B. Steux and O. E. Hamzaoui, "tinyslam: A slam algorithm in less than 200 lines c-language program," in *Control Automation Robotics Vision (ICARCV), 2010 11th International Conference on*, pp. 1975–1979, Dec 2010.

[26] W. Rone and P. Ben-Tzvi, "Mapping, localization and motion planning in mobile multi-robotic systems," *Robotica*, vol. 31, pp. 1–23, 01 2013. Copyright - Copyright © Cambridge University Press 2012; The most recent update - 2015-08-15.

[27] S. Saeedi, L. Paull, M. Trentini, M. Seto, and H. Li, "Group mapping: A topological approach to map merging for multiple robots," *IEEE Robotics Automation Magazine*, vol. 21, pp. 60–72, June 2014.

[28] S. Thrun and Y. Liu, *Robotics Research. The Eleventh International Symposium: With 303 Figures*, ch. Multi-robot SLAM with Sparse Extended Information Filers, pp. 254–266. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005.

[29] "Pioneer 3-AT Datasheet." `http://www.mobilerobots.com/Libraries/Downloads/Pioneer3AT-P3AT-RevA.sflb.ashx`, 2011. Retrieved: 2016-05-23.

[30] I. R. Mázl, *Lokalizace pro autonomní systémy.* PhD thesis, ČVUT v Praze, Fakulta elektrotechnická, Katedra kybernetiky, 2 2007.

[31] F. Hashikawa and K. Morioka, "Mobile robot navigation based on interactive slam with an intelligent space," pp. 788–789, 2011.

[32] S. Zaman, W. Slany, and G. Steinbauer, "ROS-based mapping, localization and autonomous navigation using a Pioneer 3-DX robot and their relevant issues," pp. 1–5, 2011.

[33] "SICK LMS1xx Laser Measurement Sensors - Operating Instructions." `https://www.sick.com/media/dox/1/31/331/Operating_instructions_Laser_Measurement_Sensors_of_the_LMS1xx_Product_Family_en_IM0031331.PDF`, Jul 2015.

[34] "Telegrams for Configuring and Operating the LMS1xx, LMS5xx, TiM3xx, JEF300, JEF500." `https://www.sick.com/media/dox/7/27/927/Telegram_listing_Telegrams_for_Configuring_and_Operating_the_LMS1xx_LMS5xx_TiM3xx_JEF300_JEF500_en_IM0045927.PDF`, May 2012.

[35] "ROS History [online]." `http://www.ros.org/history/`. Retrieved: 2016-05-23.

[36] "ROS Distributions [online]." `http://www.ros.org/history/`. Retrieved: 2016-05-23.

[37] J. Kramer and M. Scheutz, "Development environments for autonomous mobile robots: A survey," *Autonomous Robots*, vol. 22, no. 2, pp. 101–132, 2007.

[38] "ROS Turns 8 [online]." `http://www.ros.org/news/2015/12/ros-turns-8.html`. Retrieved: 2016-05-23.

[39] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[40] "ROS Concepts [online]." `http://wiki.ros.org/ROS/Concepts`. Retrieved: 2016-05-23.

[41] "ROSARIA - ROS Wiki [online]." `http://wiki.ros.org/ROSARIA`. Retrieved: 2016-05-24.

[42] "teleop-twist-keyboard - ROS Wiki [online]." `http://wiki.ros.org/teleop_twist_keyboard`. Retrieved: 2016-05-24.

[43] "LMS1xx - ROS Wiki [online]." `http://wiki.ros.org/LMS1xx`. Retrieved: 2016-05-24.

[44] "rviz - ROS Wiki [online]." `http://wiki.ros.org/rviz`. Retrieved: 2016-05-24.

[45] "rosbag - ROS Wiki [online]." `http://wiki.ros.org/rosbag`. Retrieved: 2016-05-24.

[46] "ROS Online Documentation - Odometry Message [online]." `http://docs.ros.org/api/nav_msgs/html/msg/Odometry.html`. Retrieved: 2016-05-23.

[47] "ROS Online Documentation - LaserScan [online]." `http://docs.ros.org/api/sensor_msgs/html/msg/LaserScan.html`. Retrieved: 2016-05-23.

[48] "gmapping - ROS Wiki [online]." `http://wiki.ros.org/gmapping`. Retrieved: 2016-05-24.

[49] "hector_slam - ROS Wiki [online]." `http://wiki.ros.org/hector_slam`. Retrieved: 2016-05-24.

[50] "stereo_slam - ROS Wiki [online]." `http://wiki.ros.org/stereo_slam`. Retrieved: 2016-05-24.

Appendix A

Contents of Enclosed CD

```
/
├── data
│   ├── BLOX_1.bag ..............first set of measured data
│   └── BLOX_2.bag ..............second set of measured data
├── documentation
│   ├── LMS111_operating.pdf ...digital version of [1]
│   ├── LMS111_telegram.pdf ....digital version of [34]
│   ├── S300_operating.pdf .....digital version of [7]
│   └── S300_telegram.pdf ......digital version of [8]
└── thesis
    └── thesis_kaposvary.pdf ..digital version of this thesis
```