# LOCALIZATION OF AN AUTONOMOUS RAIL-GUIDED ROBOT BASED ON PARTICLE FILTER

Guilherme Pires Sales de Carvalho

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Ramon Romankevicius Costa

Rio de Janeiro
Março de 2016

LOCALIZATION OF AN AUTONOMOUS RAIL-GUIDED ROBOT BASED ON PARTICLE FILTER

Guilherme Pires Sales de Carvalho

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2016

Carvalho, Guilherme Pires Sales de

Localization of an Autonomous Rail-Guided Robot Based on Particle Filter/Guilherme Pires Sales de Carvalho. – Rio de Janeiro: UFRJ/COPPE, 2016.

XV, 104 p.: il.; 29, 7cm.

Orientador: Ramon Romankevicius Costa

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2016.

Referências Bibliográficas: p. 95 – 104.

1. Localization. 2. Particle Filter. 3. Rail-Guided Robots. 4. Offshore Robotics. 5. Autonomous Robotics. I. Costa, Ramon Romankevicius. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

# Agradecimentos

Primeiramente, gostaria de agradecer aos meus pais. Sem a educação e o amor deles, eu não estaria escrevendo esta seção de agradecimentos em uma dissertação de mestrado na COPPE/UFRJ. Um agradecimento especial à Mayara, que me acompanha há mais de cinco anos com todo seu amor, apoio e compreensão nas trajetórias intensas da graduação e mestrado, mesmo do outro lado do mundo com os cangurus e coalas.

Agradeço aos amigos de graduação de ECA e aos amigos do LEAD e LABCON pelos momentos descontraídos e por ajudar a tornar o trabalho e estudo algo prazeroso. Em especial, quero lembrar meus grandes amigos que me acompanham nesta Saga ("isso mêzmo") acadêmica desde o início: Marco e Renan, a fonte e o sorvedouro de cartas. Alex, *the software guru*, deve ser citado também pelas ideias que ajudaram a desenvolver este trabalho e pela implementação de *software* para aquisição dos dados experimentais.

Agradeço à equipe do projeto DORIS, ao pessoal do SMT, aos petroleiros Deyvid e Mauricio, e à equipe de montagem do trilho no CENPES por tornarem possíveis os testes com o robô em um ambiente industrial, que recebeu até visita real.

Finalmente, agradeço aos membros da banca, professores Ramon Romankevicius, Antonio Leite e Marco Terra, pelos valiosos comentários, sugestões e críticas construtivas sobre o trabalho, mesmo compreensivamente recebendo o texto final com pouca antecedência. Em especial, tenho que agradecer ao meu orientador, chefe e amigo Ramon por me dar a oportunidade única de estudar e trabalhar com o que gosto e diretamente com ele em um projeto tão inovador e interessante. Foi, está sendo e ainda será um grande aprendizado.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

LOCALIZAÇÃO DE UM ROBÔ AUTÔNOMO GUIADO POR TRILHOS BASEADO EM FILTRO DE PARTÍCULAS

Guilherme Pires Sales de Carvalho

Março/2016

Orientador: Ramon Romankevicius Costa

Programa: Engenharia Elétrica

Recentemente, tem-se observado um crescente interesse no uso de sistemas robóticos em instalações de produção na indústria de óleo e gás, sobretudo em plataformas *offshore*. Um dos problemas nessas instalações é a manutenção de plantas de processo, que atualmente exige que operadores sejam embarcados para efetuar inspeções e intervenções no local, sendo submetidos aos riscos do ambiente agressivo característico e representando custos devidos à logística complexa.

O robô DORIS está sendo desenvolvido pela COPPE em parceria com a Petrobras e Statoil como uma solução para esse problema, tendo o objetivo de realizar tarefas de inspeção de maneira autônoma em plantas de processo no *topside* de plataformas. O robô se movimenta através de um trilho instalado em regiões de interesse na planta e é dotado de diversos sensores capazes de fornecer informações do ambiente a um operador remoto.

Este trabalho apresenta a implementação de um algoritmo de localização para o sistema de navegação autônoma do robô DORIS utilizando uma abordagem probabilística. Um filtro de partículas, que integra informações de movimentação e percepção do robô, é utilizado para estimar sua localização no trilho. Uma técnica proposta neste trabalho, baseada no histórico recente de eventos observados pelo robô, é adicionada ao algoritmo de Monte Carlo padrão para adicionar robustez, reduzir a complexidade computacional e acelerar a convergência da localização.

Simulações com dados de testes de campo com o robô mostram que o algoritmo proposto estima sua localização no trilho com erro inferior a 25cm, se mostrando superior à odometria e ao filtro de partículas padrão. Além disso, o algoritmo é capaz de resolver importantes problemas de localização para um robô autônomo: os problemas de condição inicial, localização global e o problema do robô sequestrado.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

LOCALIZATION OF AN AUTONOMOUS RAIL-GUIDED ROBOT BASED ON PARTICLE FILTER

Guilherme Pires Sales de Carvalho

March/2016

Advisor: Ramon Romankevicius Costa

Department: Electrical Engineering

Recently, it has been noted a growing interest on robotic systems in production facilities of the oil and gas industry, especially on offshore platforms. One problem in particular of those facilities is the maintenance of process plants, which currently requires flying human operators to distant fields to perform inspection and maintenance tasks on site, being subject to the risks inherent to the characteristic harsh environment and accounting costs related to complex logistics.

DORIS is a robot being developed by COPPE in collaboration with Petrobras and Statoil as a solution to this problem, with the purpose of autonomously carrying out inspection tasks in offshore platforms process plants. The robot moves through a rail installed in the regions of interest of the plant and it is equipped with various sensors capable of providing real time information about the inspected environment to a remote operator.

This work presents the implementation of a localization algorithm for the autonomous navigation system of DORIS using a probabilistic approach. A particle filter, which integrates motion and perception information of the robot, is used to estimate the robot localization on the rail. A novel technique, based on the recent history of events observed by the robot, is proposed to augment the standard Monte Carlo localization algorithm to improve the robustness and convergence rate of the estimation, and reduce its computational complexity.

Simulations using field tests data of DORIS show that the proposed algorithm estimates the robot position on the rail with an error smaller than 25cm, proving to be superior than odometry or a standard particle filter. In addition, the algorithm is able to solve important localization problems for autonomous robots, which are the initial condition, the global localization, and the kidnapped robot problems.

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# List of Abbreviations

| | |
|---|---|
| AUV | *Autonomous Underwater Vehicles*, p. 1 |
| CAD | *Computer Aided Design*, p. 9 |
| CCW | Counterclockwise, p. 50 |
| CENPES | *Centro de Pesquisas da Petrobras*, p. 7 |
| DBN | coating thickness measurement, p. 14 |
| DoF | Degree of Freedom, p. 8 |
| EKF | *Extended Kalman Filter*, p. 15 |
| GNSS | *Global Navigation Satellite Systems*, p. 11 |
| GPS | *Global Positioning System*, p. 15 |
| HER | *History of Events Resetting*, p. 11 |
| HSE | *Health, Safety and Environment*, p. 3 |
| HSV | *Hue, Saturation and Value*, p. 56 |
| IF | *Information Filter*, p. 27 |
| IMU | *Inertial Measurement Unit*, p. 10 |
| INS | *Inertial Navigation System*, p. 15 |
| KF | *Kalman Filter*, p. 23 |
| MCL | Monte Carlo Localization, p. 43 |
| MVN | *Multivariate Normal Distributions*, p. 23 |
| PDF | *Probability Density Function*, p. 23 |
| PF | Particle Filter, p. 28 |

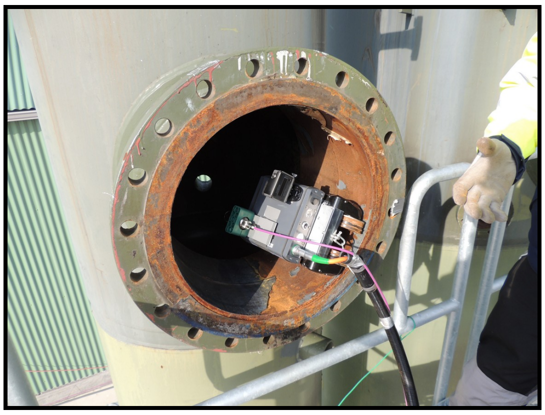
| | |
|---|---|
| PIG | Pipeline Inspection Gauges, p. 1 |
| RGB | *Red, Green and Blue*, p. 56 |
| ROS | *Robot Operational System*, p. 7 |
| ROV | *Remotely Operated underwater Vehicle*, p. 1 |
| RPY | *Roll, Pitch and Yaw*, p. 50 |
| RoI | *Region of Interest*, p. 63 |
| SLAM | *Simultaneous Localization And Mapping*, p. 15 |
| SRL | *Sensor Resetting Localization*, p. 37 |
| UKF | *Unscented Kalman Filter*, p. 27 |

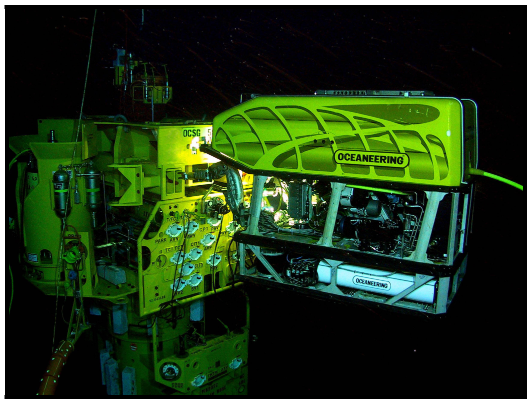
# Chapter 1

# Introduction

Traditionally, robotics in the oil and gas industry is mainly applied in inspection of underwater equipment and seabed mapping by *Remotely Operated underwater Vehicles* (ROV) and *Autonomous Underwater Vehicles* (AUV) , as well as in pipeline, tank and vessel inspection (Bos et al. 2015, Faber Archila & Becker 2013, Shukla & Karki 2013, Yuh 2000). Some examples of robots used in these applications are shown in Figure 1.1.

(a) A Pipeline Inspection Gauge (PIG).


(b) PETROBOT's vessel inspection robot.



(c) An ROV operating on a subsea structure.

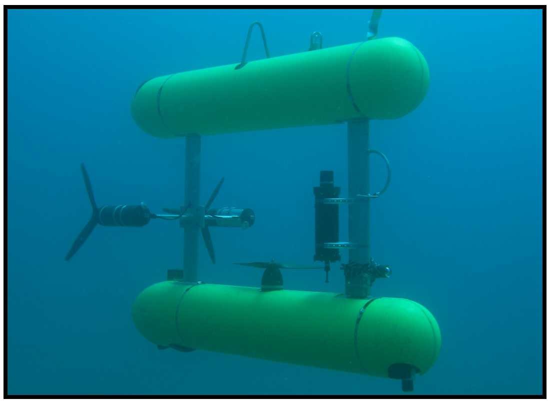

(d) An AUV mapping the seabed.

Figure 1.1: Examples of current robotic applications in the oil and gas industry.

The motivation for the use of robotics in the oil gas industry comes from the repetitive, unhealthy and hazardous tasks characteristic of oil and gas facilities. In view of these conditions, such type of work should not be performed by human beings, specially when the new oil fields in deepwater are considered. Furthermore, the robotics technology employed in the applications cited above is developed in a level that the required robustness and cost-effectiveness are provided.

On the other hand, the same attention has not been given to robotized systems on the topside of platforms, mainly because this technology still has not shown to be robust and safe enough, and with the desired cost-effectiveness for the current oil exploration fields. However, the recent advances in robotics in the areas of autonomy, teleoperation, and non-structured environments, combined with the inherent challenges of future oil fields are changing this scenario and motivating research and development of robots for the topside of offshore platforms.

Currently, the main variables of a process plant in a platform are monitored through instrumentation and automation networks. The installation and maintenance costs of several sensors are proportional to the dimension of the process plant. Moreover, studies show that the complete automation of an oil and gas facility requires a system with 1000 operations additional to those already carried out by operators (Anisi et al. 2011), such as valve manipulation, sample taking, and insertion and removal of PIGs. A mobile robot provided with a variety of sensors concentrated in just one place can dramatically reduce the number of fixed sensors in an automation network, and, consequently, the costs and risks associated to the installation and maintenance of these sensors (NREC/CMU 2012).

The depletion of oil resources within easy reach is imminent and an increase in the current oil price may turn the exploration of previously unprofitable oil fields feasible. Future exploration fields are more remote and have more severe environmental conditions, representing challenges that require new technologies and even new business models, which tend to increase the level of automation in platforms and decrease human activity in such environments (Skourup & Pretlove 2009). New technology will allow, besides of the exploration of challenge oil fields, the extension of the current platforms life cycle. The trends are that future offshore platforms will be completely uninhabited (Pfeiffer et al. 2011).

The higher costs associated with logistics and production of oil exploration in remote and deepwater areas demand higher productivity and robustness. From (2010) highlights that robotic operators are productive, given that they work 24 hours a day and 7 days a week, are more precise, and less prone to errors. Thus, one can expect a smaller number of failures and unplanned production downtime, also minimizing commissioning time and the risks of accidents (Johnsen et al. 2007).

Another important advantage in the use of robotic solutions is the improvement on Health, Safety, and Environmental (HSE) conditions, which has been receiving a growing concern by oil and gas organizations. Several tasks performed by humans in unhealthy, harsh, and confined areas, which are characteristic of an offshore facility, as shown in Figure 1.2, could be assigned to robotic operators.

Figure 1.2: Operators in harsh conditions, typical of an offshore environment.

A feasibility study mapped tasks on the topside of offshore platforms that could be performed by robots, implying a reduction of the operational costs and human exposure to HSE risk factors. The research concluded that human labour in offshore platforms operation can be reduced by 50%, and the number of planned manual interventions can be limited to two times a year with robotic technologies (Pfeiffer et al. 2011). Vatland & Svenes (2008) points out that automation in oil rigs decreases the capital expenditures by 30% and the operational costs by 32% .

The employment of robots in offshore environments brings many challenges to be overcome. Temperature in oil facilities ranges from -30°C to 50°C, relative humidity can reach 100%, there may be splashing water, salt spray, storms, and severe corrosion with toxic gases (Graf & Pfeiffer 2007, Skourup & Pretlove 2009). Furthermore, the robot must meet the required safety standards to operate in classified areas.

Another difficulty to be highlighted is related to autonomous navigation. Autonomous robots are generally able to detect walls, surfaces and obstacles to localize themselves and plan trajectories to avoid collision. However, offshore platforms contain complex structures in cluttered areas, such as pipelines, and balustrades as limiting boundaries. These characteristics are difficult to be detected by typically employed sensors in autonomous navigation, such as a laser range finder (Graf & Pfeiffer 2008). Moreover, the platform floor contains steel gratings, and some of its areas may be divided by levels with stairs, which hampers the mobility of the robot.

The idea of using mobile robots on the topside of offshore platforms was introduced in P. Liljebäck & Schumann-Olsen (2005). However, the first implementation took place with MIMROex robot (Bengel & Pfeiffer 2007), developed by Fraunhofer IPA, in Germany. MIMROex has a two-wheeled base and a robotic arm with a camera attached to its end-effector (Figure 1.3). The robot is certified to operate in classified areas and has a navigation system that uses a laser scanner to detect poles and reflective strips for self-localization, mapping of the surrounding environment, and obstacle avoidance. According to Bengel et al. (2009), the robot proved to be robust to different environmental conditions, being able to safely navigate, map the environment, and execute inspection tasks autonomously. This first implementation has shown the feasibility of using mobile robots on the topside of offshore platforms.

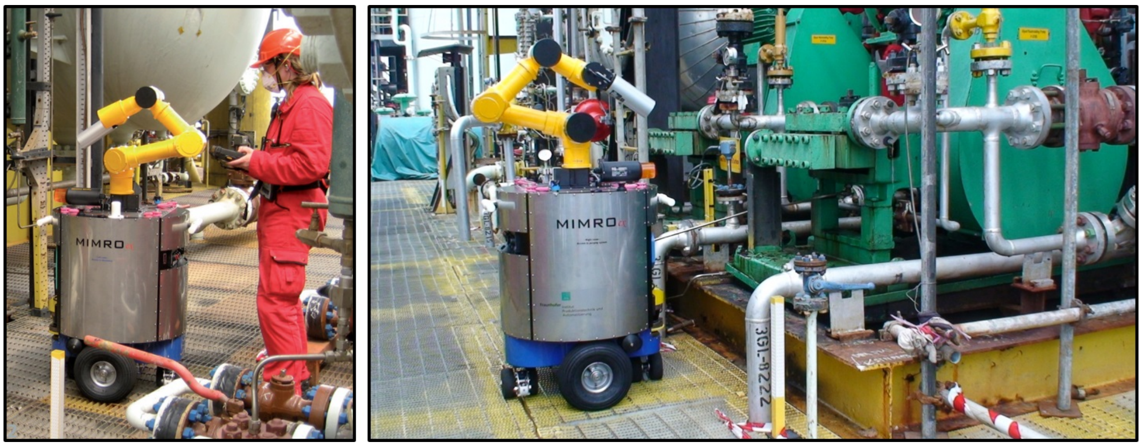


Figure 1.3: MIMROex being tested on an offshore facility. (Bengel & Pfeiffer 2007)

Another mobile robot already tested on an oil facility is Sensabot (Figure 1.4), developed by the National Robotics Engineering Center (NREC) of Carnegie Mellon University (CMU). It is a four-wheeled device with cameras, gas sensors, a vibration sensor, a microphone, a laser scanner, and a robotic manipulator (NREC/CMU 2012). The robot is certified to perform inspection and monitoring tasks in flammable, explosive and toxic environments. The complete system also includes a mechanism that allows the robot to access different platform levels by attaching its wheels to a cog rail.

In addition to mobile robots, the use of industrial manipulators for operation in platforms has also been recently studied. A research group involving ABB, Shell

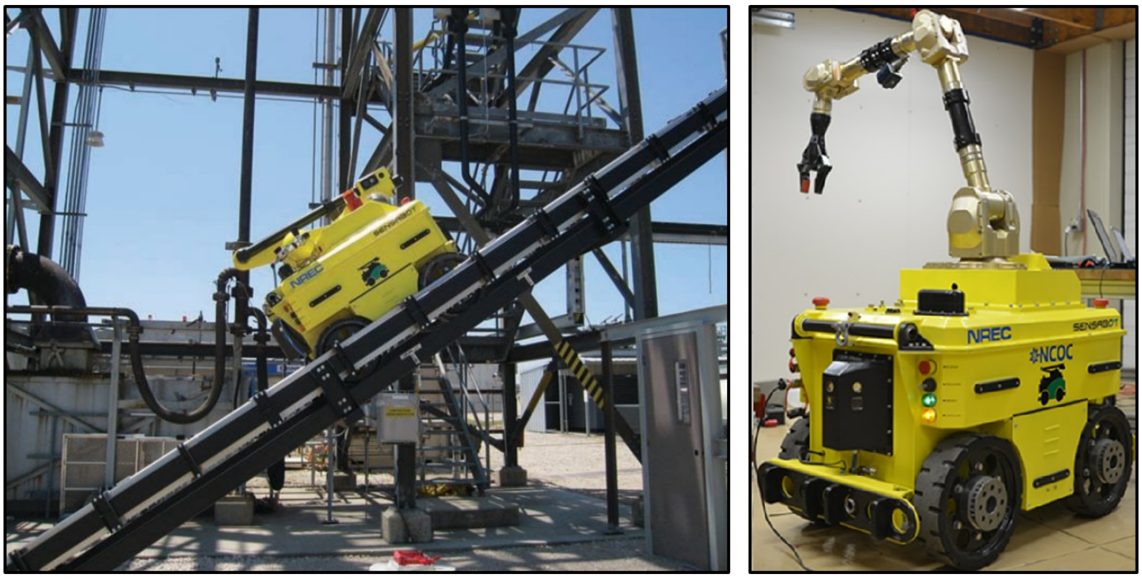


Figure 1.4: Sensabot being tested in an offshore environment. (NREC/CMU 2012)

and Statoil adopts a step-wise strategy to demonstrate and validate technologies for offshore applications (Anisi et al. 2012). The research focuses on valve manipulation, an operation that represents several challenges to robots.

In a first moment, proofs of concepts are carried out in a controlled laboratory environment (Anisi et al. 2010). As an intermediate step, the technology is validated on an external facility (Anisi et al. 2011). Finally, the system is tested under on-site operational conditions, that has safety requirements and a severe environment. These steps are shown in Figure 1.5.

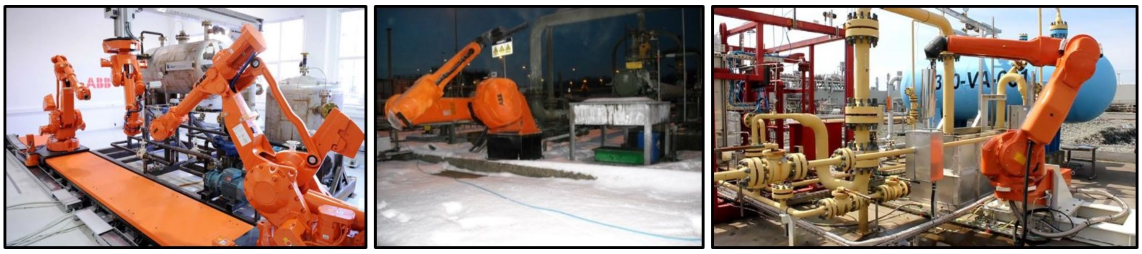


Figure 1.5: Industrial manipulators being tested in a lab and then in real operational conditions. (Anisi et al. 2012, 2010)

SINTEF-ICT, together with Statoil and NTNU, forms another research group interested on the development of robotic technologies for offshore platforms. They present a new concept for remote inspection and maintenance with the division of the platform workspace between an accessible area to humans and a permanently inhabited area, which has only robots (Figure 1.6). The research group studies the cooperation between a gantry-mounted manipulator and a floor-mounted robotic arm on a simulated process plant (Bjerkeng et al. 2011, Fjerdingen et al. 2012, Kyrkjebø et al. 2009, Transeth et al. 2010).

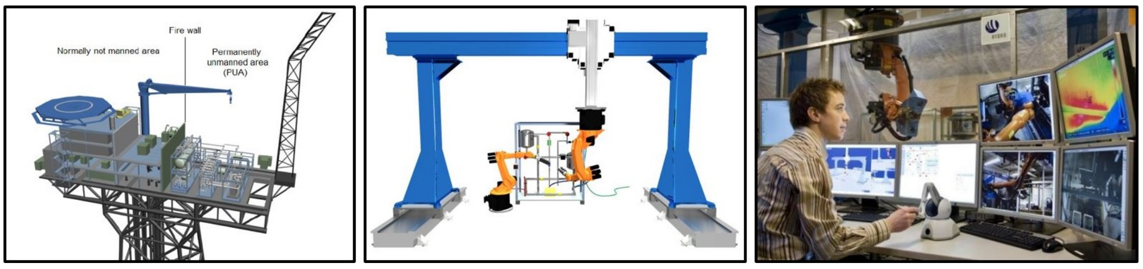


Figure 1.6: SINTEF-ICT futuristic concept for offshore platforms with cooperative manipulators. (Fjerdingen et al. 2012, SINTEF 2008)

Another recent highlights in onshore and offshore robotics research are the PETROBOT project (Bos et al. 2015), carried out by the energy company Alstom together with ETH Zürich, which develops modular robots for tank inspection, and the ARGOS challenge, organized by the oil and gas company Total, which will be completed in December 2016 (Kydd et al. 2015). More information about the state of the art in offshore robotics is found in Transeth et al. (2013).

## 1.1 DORIS Project[1]

Huge oil reserves in the pre-salt layer of the Brazilian coast were recently discovered. These oil reservoirs are located farther than 300 km from the shore and at 5000m to 8000m below the sea level. The economic value and the challenges of exploration in remote and deepwater fields motivates the development of an offshore production system with a high degree of automation based on advanced robotics.

COPPE/UFRJ, in collaboration with Petrobras and Statoil, has been developing technologies for offshore robotics. In addition to works in the areas of virtual reality for robotic teleoperation (Carvalho et al. 2014, Santos et al. 2013) and autonomous manipulation of valves with robotic arms (Faria et al. 2015), the research group is developing the DORIS robot (Figure 1.7).

DORIS is a rail-guided robot conceived to perform monitoring and inspection tasks on the topside of offshore platforms (Carvalho et al. 2013) carrying several sensors and a robotic arm. The system can identify anomalies in the operation through its following functionalities:

- **Video**: detection of video anomaly, visualization of control panels, level indicators, switch positions, and LED status through cameras;
- **Audio**: detection of audio anomaly, and monitoring of rotating machines noise pattern with microphones;

[1]This project is supported primarily by Petrobras S.A. and Statoil Brazil Oil & Gas Ltda under contract COPPETEC 0050.0079406.12.9 (ANP-Brazil R&D Program), and in part by the Brazilian research agencies CNPq and FAPERJ.

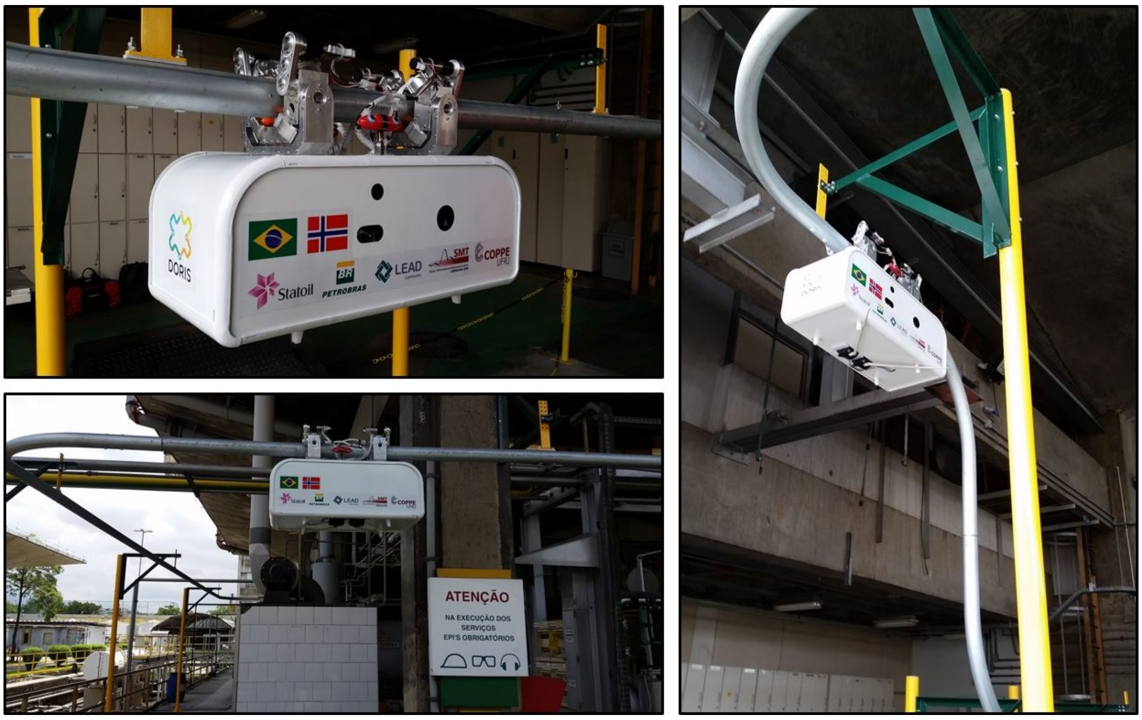


Figure 1.7: DORIS being tested in an industrial environment.

- **Temperature**: monitoring of the thermal pattern of equipments, and detection of pipeline blockages and leakages with a thermal camera;
- **Vibration**: machinery vibration diagnosis through a vibration sensor attached to the manipulator arm;
- **Robotic manipulator**: assistance to the operator in the visualization of the process, and possible interaction with it through sensors and actuators attached to the end-effector.

The robot is controlled autonomously or by teleoperation, providing to the operator online access to the embedded sensors and real time information about the monitored environment and the robot operating conditions using the Robot Operational System (ROS) (Quigley et al. 2009) as the operational system. DORIS weighs approximately 30kg and moves with a maximum speed of 1m/s.

The system concepts of the traction mechanism, rail type, and teleoperation and telemetry capabilities were firstly validated in a laboratory, as well as preliminar results of the video and audio processing algorithms (Freitas et al. 2015, Galassi et al. 2014). Currently, the robot is being tested in a utility plant at CENPES , a Petrobras research facility, to demonstrate its functionalities in an industrial environment as an intermediate step to a further implementation on offshore platforms.

## 1.2 Problem Statement

DORIS moves in the monitored environment through a specifically designed rail, composed of straight and curved tubular segments (Figure 1.8). The robot can be positioned anywhere on the 3-D rail, including on vertical sections, by means of friction between its wheels and the tubular rail, and by a complex patented traction mechanism. The wheels are comprised in gimbal mechanisms that suspend the robot on the rail and provide the necessary *Degrees of Freedom* (DoF) for 3-D motion.

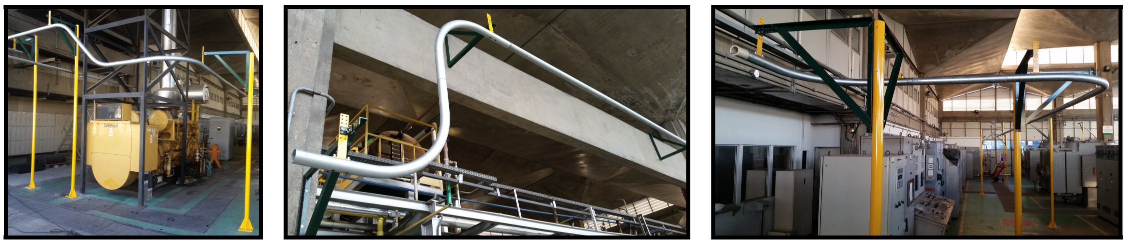


Figure 1.8: Installation of DORIS rail system in a utility plant in CENPES.

The use of a rail as the robot locomotion mechanism alleviates many difficulties of autonomous navigation, such as mapping, obstacle avoidance, motion on different terrains, path planning, and localization. However, when complete autonomy is desired for a robotic system, robustness is essential. The system must be resilient to unpredictable or sudden changes of its operational conditions, such as failures, and has to take decisions autonomously (Freitas 2016).

Furthermore, specific tasks of DORIS, such as vibration measurements with the robotic arm in a particular area of the monitored environment (Xaud 2016), audio and video alignment in signal processing algorithms, and switching of traction modes on the rail, demand high accuracy, so that the localization problems are not completely eliminated. As an example of how poor odometry can affect the robot functionalities, consider an error of 1% in odometry on a rail of 1Km of extension. The accumulated error in localization would be of 10m after moving throughout the total rail extension, which is unacceptable.

The movement of a rail-guided vehicle is predictable and constrained, which, at first, enables self-localization by simple odometry. However, this system does not show to be robust to variations on the operational conditions and neither is accurate enough for specific tasks, substantially after long periods of motion.

The major problem of an odometric system in a mobile robot is admittedly the wheel slippage. Even though the robot is constrained to the rail, slipping is inevitable due to the traction being provided by friction between the wheels and the tubular rail surface. This is specially true in vertical motion or when the rail surface is slippery due to rain, salt spray, or grease. Another problem associated with odometry is that it requires the complete knowledge of the system parameters, such as the wheel diameter.

However, maybe the most significant problem addressed by advanced localization techniques is the initial condition uncertainty. An odometric system works only if the robot initial position is known, which is a strong assumption from the practical point of view. Advanced localization techniques relax or even dismiss this hypothesis, dealing with partially known (*tracking problem*) or completely unknown (*global localization problem*) initial conditions.

An even more complex problem that can be addressed by some localization methods is known as the *kidnapped robot problem* (Engelson & McDermott 1992). In this case, it is considered that the robot firmly believes it knows where it, is while it actually does not. Eventual failures in the robot system that reset or keep its states constant while moving during the time of failure are examples that can cause this situation. The kidnapped robot problem is often used to test the robot ability to autonomously recover from catastrophic localization failures (Fox et al. 2001).

The problem to be addressed in this work is basically to reliably localize DORIS on the rail, regardless of the knowledge of its initial conditions or even after occasional failures. In other words, the objective of this work is to solve the tracking, global localization, and kidnapped robot problems for DORIS.

## 1.3 Proposed Solution

To deal with the tracking, global localization and the kidnapped robot problems for DORIS, a particle filter approach is considered. This type of probabilistic filter was chosen due to its natural ability of dealing with multiple hypothesis on the state estimation, and solving complex localization problems with an easy implementation (Thrun et al. 2005). Given a known map of the rail, DORIS motion and perception information through time is integrated in the filter to compute the belief of the robot, which is its possible states and how probable these states are of being true.

As DORIS rail is specifically designed for it, a *Computer Aided Design* (CAD) model is available, and thus, the rail map is supposed to be known. This assumption is not completely true, given that there is uncertainty in the fabrication of the rail segments, and mainly in the rail installation. As an example, consider a 90° curved segment followed by a 50m straight track. An installation error of 1° in the joint between the two segments results in a side error of almost 1m, preventing the robotic arm to reach a previously considered workspace, for example. However, probabilistic approaches to state estimation problems, as particle filtering, deals with several types of uncertainty, including those of the map model.

A special feature of a rail-guided robot is that the rail constrains the robot motion, making it have only one DoF, which is going forwards or backwards on the along-track direction of the rail. Due to this motion constraint, the robot gimbals poses (cartesian positions and orientations) are equivalent to the rail poses (Figure 3.6). Therefore, the localization problem is limited to find the robot in a digital map of the rail that provides these poses, and not in the full continuous world.

Another particular characteristic is that the rail is composed of segments that are one-dimensional elements. Therefore, a single variable $s$, which parameterizes the length of the traveled track, is enough to determine the rail poses and, consequently, the robot poses. Thereby, the robot can be completely localized by determining this variable, which is considered as the only state variable to be estimated in the localization problem of DORIS.

The classical solution of probabilistic state estimation applied for robotics is to combine a prediction step given by the robot motion model with a correction step of the estimate after receiving environment perception information through multiple sensors. In this implementation, the motion model is considered to be the state transition of the variable $s_t$ at a time $t$, given by the odometric equation:

$$s_t = s_{t-1} + v_t \Delta T \,, \tag{1.1}$$

where $v_t$ is the velocity of the robot at the time $t$, used here as the control input, and $\Delta T$ is the difference in time between two steps of the algorithm. The robot velocity is obtained by scaling the motors rotary encoders information by a factor given by the transmission ratio and the wheel diameter. As all the four motors have the same velocity control setpoint, the mean of the four encoders information is used. In a future improvement, the switching of motion modes will be considered, as the wheels velocities in curved parts should be individually controlled.

Another great advantage of rail-guided robots is that the natural features of the rail or artificial landmarks placed on known positions can be recognized by the robot and used in localization. Therefore, DORIS uses a monocular camera to detect red markers placed on known locations of the rail, and a laser scanner to estimate the robot distance to the floor and detect natural features of the rail, such as the rail fixation system and the rail geometry. Additionally, an I*nertial Measurement Unit* (IMU) provides attitude information.

Even though some of these measurements are not given in the estimation state space, they are mapped through it by:

$$z_t = f_{map}(s_t, m), \tag{1.2}$$

in which $z_t$ is a measurement at a time $t$ and $f_{map}$ is a known mapping function, which is derived in Chapter 3. The motion model, based on (1.1), and the measurement models, based on (1.2), are integrated in a particle filter to estimate the variable $s_t$ through time. The particle filter resampling step is taken based on an estimation of the effective number of particles. Further, a novel extension to the standard particle filter implementation, named *History of Events Resetting* (HER), is employed to improve the estimation, enabling the solution of the global localization and the kidnapped robot problems with a small number of particles.

Simulation results using data acquired in field tests show that the proposed algorithm addresses the three types of localization problems mentioned above (tracking, global localization, and the kidnapped robot problems) with a precision of approximately 25cm and using only a small number of particles (50), proving its advantages over odometry and the standard particle filter.

## 1.4 Literature Review

In this section, the main sensing techniques employed in mobile robot localization problems are reviewed. The principles of Bayesian filters, which are the basis of the state of the art in probabilistic filtering, are briefly introduced before a detailed study in Chapter 2 along with the main probabilistic localization algorithms. Then, some applications of localization techniques for rail-guided vehicles are presented.

The common approach in mobile robot localization is to integrate information from an odometric motion model with multiple sensors data in a Bayesian filter. The choice of which types of sensors to use and how to process the sensed data plays a big role in the state estimation problem. Sensors can be distinguished into two main groups: intrinsic robot sensors and extrinsic environment perception sensors.

Intrinsic sensors are related to proprioception, or the ability to sense the robot own internal states. Odometers, wheel encoders, and inertial measurement units are examples. Methods based only on intrinsic sensors, which are known as relative localization or dead-reckoning, have several implementations (Angermann & Robertson 2012, Chitta et al. 2007, Doh et al. 2006, Ndjeng et al. 2008, O'Kane 2006). A drawback of these systems is that they integrate relative increments, and errors can considerably grow over time due to parameter uncertainty or bias (De Cecco 2003).

A more robust approach is to integrate these measurements with extrinsic sensor data, which provide absolute information about the surrounding environment. A vast number of work has been done in localization problems using this type of sensor, which commonly are cameras, range finders (sonars and lasers), and *Global Navigation Satellite Systems* (GNSS). Other not so common technologies applied to mobile robot localization are *Radio-Frequency IDentification* (RFID) (Hahnel et al. 2004), *Wi-Fi* (Biswas & Veloso 2010), and magnetic field sensing (Christensen, Fischer, Kroffke, Lemburg & Ahlers 2011, Robertson et al. 2013).

### 1.4.1 Vision-based Localization

The use of vision in mobile robot navigation has been the source of countless research contributions. Usually, image information is used to detect and possibly learn natural landmarks (Thrun 1998), as plans, corners, and edges in indoor environments, or artificial landmarks, such as known patterns (barcodes, QR codes, ...), and colored geometric objects or structures. The detected landmarks are then compared to a database of known landmarks in a map of the environment that gives information about the robot current localization.

Another approach to estimate position and orientation through image is *visual odometry* (Nistér et al. 2004), which consists of estimating the camera motion from optical flows composed of vectors that represent the displacement of detected features in the image sequences. A recent survey of vision-based algorithms for localization and mapping problems is provided in Bonin-Font et al. (2008).

### 1.4.2 Scan-based Localization

A common device used in mobile robots is a range finder. Sonars, mainly used in underwater applications, and laser scanners measure range and bearing from objects hit by the emitted beam. One approach to provide localization information from range and bearing scans is to track the robot position relative to known landmarks, as geometric beacons (Leonard & Durrant-Whyte 1991). These features can be identified in the 2D scan trough methods such as the *Random Sample Consensus* (RANSAC) or the *Hough-Transform* algorithms. A complete comparison of line extraction methods in laser scans is found in Nguyen et al. (2005).

Another possibility is to compare the entire 2D scan (Diosi & Kleeman 2005) or some of its recognized features (Aghamohammadi et al. 2007, Nieto et al. 2007) to a known point cloud. The best matching between the target and reference scans provides the position and orientation of the robot relative to an inertial frame. This technique is called *scan-matching* and is generally applied using the *Iterative Closest Point* (ICP) algorithm (Besl & McKay 1992), where, for each point in the target scan, the point with the smallest *Euclidean distance* in the reference scan is selected.

### 1.4.3 IMU and GNSS-based Localization

Inertial measurement units are usually composed of three accelerometers and three gyroscopes, which measure, respectively, linear accelerations and angle rates about the three axes of the sensor coordinate system. *Micro-Electromechanical Systems* (MEMS) gyroscopes are the most available ones, and features a bias of about $3°/h$ to $20°/h$, so that additional sensing is necessary to bound the error.

The gravity estimation by the accelerometers is used to derive the body inclinations about the axes perpendicular to the gravity direction, which are the *roll* and *pitch Euler angles*. To compute the heading (or *yaw*), additional sensing is needed, which usually is provided by magnetometers. Magnetometers measure the Earth's magnetic field to find the North Magnetic Pole, but its accuracy ranges from $0,5°$ to $10°$ (Advanced Navigation 2013), and it is subject to static and dynamic magnetic disturbances caused by the robot itself or by nearby ferromagnetic structures.

Bearing this in mind, GNSS systems are generally integrated with IMUs in navigation problems, mainly in outdoor environments, as for cars (Montemerlo et al. 2008, Thrun et al. 2007), trains (Heirich et al. 2013), and wheeled robots (North et al. 2012). GNSS provides absolute geospatial position and velocity with global coverage, given that at least four satellites are in reach of the GNSS receiver.

The usual approach is to compensate odometry and inertial measurement errors with GNSS data, when available. However, it is common to have GNSS outages due to clouds, trees, and cluttered environments blocking the GNSS signal. Moreover, its typical position error ranges from 2m to 8m (Advanced Navigation 2013), making the estimation too rough for some applications. In this case, the system must be augmented with more accurate sensors.

### 1.4.4 Bayesian Filters and Markov localization

Bayesian filters are the bases of the probabilistic approach to robot localization and mapping problems. They can be understood as a recursive method that integrates a prediction step, given by the robot motion, with a correction step, provided by the robot perception of the environment, to compute state estimations in the form of probability densities.

Usually, Bayesian filtering techniques are simplified by considering the *Markov assumption*, which postulates that past and future data are independent if one knows the current state. The state is said to be complete, which means that the knowledge of past states, measurements, or controls carries no additional information (Thrun et al. 2005). Probabilistic algorithms for state estimation are also referred to as filters in the sense that they provide a better estimation than the robot measurements or predictions alone. The current state-of-the-art methods are detailed in Chapter 2.

### 1.4.5 Rail-guided vehicle localization

Regarding the localization problem of DORIS, the most similar works found in the literature are related to the localization of rail-guided vehicles and robots, which are basically 1-D localization problems where the robot motion is constrained by the rail geometry. Even though the localization problem for rail-guided vehicles is attenuated, all of these related works use advanced techniques based on probabilistic filters to deal with uncertainty and ambiguity in the state estimation.

ARTIS (*Autonomous Rail-guided Tank Inspection System*, Figure 1.9) is a ballast water tank inspection robot, developed by the DFKI Research Center, in Germany. The robot is composed of modules with different sensors and moves on a thermoplastic rail. According to Christensen, Fischer, Kroffke, Lemburg & Ahlers (2011), the magnetic field of the inspected environment is static and shows particularities that can be used for localization purposes. Taking advantage of this feature, the robot integrates odometry information from the motor encoder with gravitational and magnetic data from an IMU in a particle filter to estimate its localization.

CAD data provides a geometric map, while the magnetic and gravitational map are built after cycling through the rail with the IMU. The accuracy of the localization system is enhanced by a computer vision template matching algorithm with the camera images to detect artificial landmarks distributed on the rail (Christensen, Kirchner, Fischer, Ahlers, Psarros & Etzold 2011).

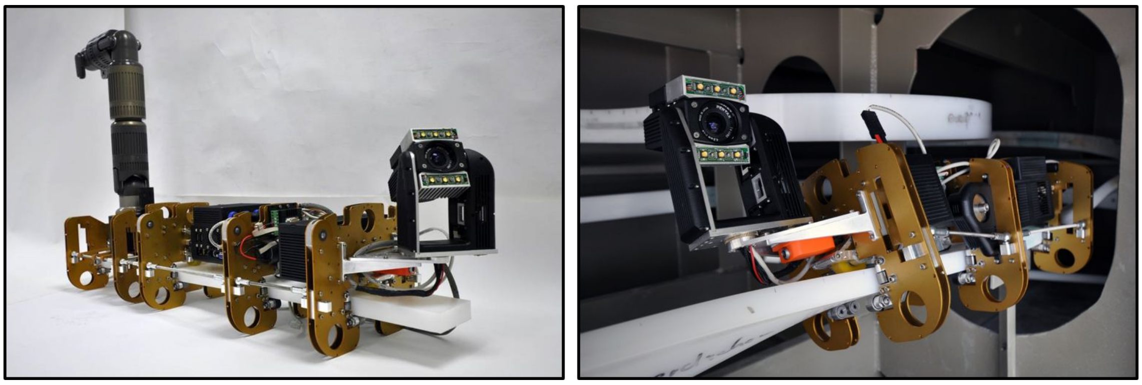


Figure 1.9: ARTIS carrying sensors and a robotic arm in different modules on a thermoplastic rail. (Christensen, Kirchner, Fischer, Ahlers, Psarros & Etzold 2011)

As pointed in Borgerink et al. (2014), the robot has one specific task that requires a high position accuracy, just like DORIS has, which is the coating thickness measurement with an embedded manipulator. The study points out that, due to ship motion and the natural compliance of the plastic rail, position errors arise and have to be compensated.

The Railway Collision Avoidance System (RCAS) is a project led by the German Aerospace Center (DLR) that aims to develop technologies for rail collision avoidance in European railways. The group implements probabilistic localization and mapping algorithms based on *Dynamic Bayesian Networks* (DBN) using onboard train sensors to achieve precise localization in the topological map of the rail network.

In Heirich et al. (2012), a particle filter is proposed, where the train motion is modeled as a *Discrete Wiener Process Acceleration* (DWPA) (Bar-Shalom et al. 2004)) and the train states are given by a mapping function of the topological pose of the train and its displacement given on the variable that parameterizes the track length. GNSS and IMU measurements are used to compute the particle weights. Rail feature classification sensors, such as switch way detectors based on vision or eddy current (Hensel et al. 2011), are suggested to improve the robustness of the localization system.

The problem is further extended to *Simultaneous Localization And Mapping* (SLAM), where the railway map is supposed to be unknown. *RailSLAM*, as named by the authors, is implemented in García (2012) as a particle filter that uses GNSS and IMU measurements, where each particle carries individual maps and a global map is computed periodically. The rail is parameterized as a piece-wise linear polynomial interpolation.

In Heirich et al. (2013), an *Extended Kalman Filter* (EKF) is used to estimate the train topological pose, geographic position, and the rail curvatures. According to the authors, this work is an intermediate step to a future *Rao-Blackwellized SLAM* implementation, as is done in Montemerlo et al. (2002) and Montemerlo & Thrun (2007). In this case, the particles rely on multiple realizations of the map, while Gaussians represent the internal vehicle states for each particle.

The localization and mapping of rail vehicles is also studied by the Karlsruhe Institute of Technology (KIT). In Hensel & Hasberg (2010), the authors argue that odometry and GNSS are insufficient for an accurate localization and use only an *Eddy Current Sensor* (ECS) to estimate the train velocity and the traveled distance on a rail by a rail sleeper counting method. The ECS is also used to detect natural rail features, such as turnouts and bridges, to find the vehicle in the known topological map. In Hensel et al. (2011), a particle filter is implemented to resolve ambiguous hypotheses about the train position after passing through a rail turnout.

The KIT research group also investigates the SLAM problem for path-constrained motion vehicles in Hasberg et al. (2012). A 1-D representation of the vehicle kinematics in arc-length coordinates of local parameterized cubic spline curves is used to compute an EKF. The method is validated in a train positioning system using an *Inertial Navigation System* (INS) and *Global Positioning System* (GPS).

## 1.5 Text Organization

This dissertation is organized as follows:

**Chapter 2** presents a deep review of the state of the art probabilistic techniques for mobile robot localization. The main algorithms are detailed with discussions about advantages, limitations, and possible solutions to attenuate or even overcome these limitations.

**Chapter 3** addresses the interaction between DORIS and its environment. A known map is created from a CAD model in a way that all the rail features are accessed from a single parameter. Motion and perception models of the robot are derived to be integrated in the particle filter.

**Chapter 4** shows the particle filter implementation for DORIS localization and the results of simulations with data acquired in field tests. A novel extension to the standard particle filter is proposed in order to overcome some of the standard particle filter limitations.

**Chapter 5** presents the final remarks, and future works are suggested.

# Chapter 2

# State of the Art in Localization

This chapter is based on contents of the book *Probabilistic Robotics* (Thrun et al. 2005). Some techniques presented there will be restated here for self-containess.

Probabilistic robotics is a relatively recent area that deals amazingly well with uncertainty, which is a natural characteristic of several robotic systems. It can be traced back to the Kalman filter (Kalman 1960), but only in the last two decades it has received more attention, due to the increasing number of mobile robots, which are less predictable than fixed robotic manipulators, for example. Mobile robots have to deal with several types of uncertainty, as unmodeled variables, unstructured environments, and noisy sensors. Therefore, managing uncertainty is perhaps the most important step towards robust real-world systems (Thrun et al. 2005).

The key advantage of a probabilistic approach for state estimation is that it represents uncertainty through time using the *probability theory*. While a deterministic method considers that the state is exact, a probabilistic approach returns estimates and values that represent how probable of being true are these estimates. This information is much valuable, as the robot can, for example, work with multiple hypothesis and even take decisions based on how certain it is about its state, as in *active localization*, which is a category of *exploration* where the robot seeks to minimize its uncertainty (or maximize its knowledge about the external world) to achieve self-localization (Fox et al. 1998).

## 2.1 Definitions

To derive the probabilistic algorithms presented in this chapter, one must consider the robot interaction with its environment. An environment is a dynamic system that possesses internal state, and the robot interaction with it can be categorized in two different ways: motion and perception.

Robot motion is the influence on the environment caused by the robot actuators, which affects both the environment state and the robot internal belief with regards to this state. Motion in mobile robotics is somewhat unpredictable, and, generally, makes the robot less certain about its belief of the true state.

Perception is the robot's ability to acquire information from the environment using its sensors. Even though sensors may be noisy and there are many variables which cannot be measured directly, the perception process usually makes the robot more confident about its belief.

The localization problem consists of integrating motion and perception iteratively, given a map, to provide a state estimation. When the map is unknown, the problem is extended to the *Simultaneous Localization and Mapping* (SLAM) problem. The SLAM problem is of great importance in mobile robotics, as usually one does not have complete knowledge of the map a priori, or the environment is unstructured. A great review of state of the art SLAM algorithms is found in Durrant-Whyte & Bailey (2006) and Bailey & Durrant-Whyte (2006). In the present work, the map is supposed to be known, and SLAM algorithms are not implemented.

To understand the algorithms derivations in this chapter, one needs to establish the following important definitions about the robot interaction with its environment:

- **State**: the state vector at a time $t$, denoted here as $x_t$, is the collection of all variables of the robot and its environment that can impact the future. States can be static or dynamic when they change with time. Thus, the vector state has to be updated on every instant $t$ with the measurements and the control actions. Examples of state variables are the robot pose and the state of objects in the environment, such as an open or closed door.

- **Control Actions**: the control actions at a time $t$, which are represented here as $u_t$, change the state of the robot and the environment. Examples of control actions are the robot motion and the manipulation of objects.

- **Measurements**: measurements are the result of a perceptual interaction of the robot with the environment through its sensors. The group of all measurements taken at a time $t$ are denoted here as $z_t$. Examples of measurements are laser scans and visual detection of landmarks with a camera.

- **Map**: the map, referred in this work as $m$, is the representation of the environment, generally given by sets of landmarks, which can influence the robot motion and perception. The map may be supposed to be known or unknown, which, in the latter case, can be included in the state vector to be estimated.

## 2.2 Probabilistic Generative Laws

In probabilistic robotics, the evolution of states, control actions, and measurements is governed by probabilistic laws. At first, the state $x_t$ might be conditioned on all past states, control actions, and measurements. So, at a discrete time $t$, the state $x_t$ is stochastically calculated from the previous states $x_{0:t-1}$, control actions $u_{1:t}$, and measurements $z_{1:t-1}$ [1] by the probability distribution $p(x_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t})$.

However, a reasonable approximation for most mobile robotics applications is to have *state completeness*, in which the state $x_{t-1}$ encompasses the information of all the measurements $z_{1:t-1}$ and control actions $u_{1:t-1}$ that happened in previous time steps. In this case, the process is said to satisfy the *Markov property.* This approximation allows the expression of the posterior probability of the state $x_t$ by:

$$p(x_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t}) \overset{\text{Markov}}{=} p(x_t \mid x_{t-1}, u_t)\,, \tag{2.1}$$

where $x_t$ is conditioned only on the previous state $x_{t-1}$ and the most recent control action $u_t$. This is called the *state transition probability* and specifies how environmental state evolves over time as a function of the robot control $u_t$.

It is also important to model the process by which measurements are being generated. The measurement model can be defined to describe the probability of generating a certain observation conditioned on the previous states, measurements, and control actions. Under the same assumption of state completeness of $x_t$, the probabilistic measurement model, which is commonly called *measurement probability*, is given by:

$$p(z_t \mid x_{0:t}, z_{1:t-1}, u_{1:t}) \overset{\text{Markov}}{=} p(z_t \mid x_t) \tag{2.2}$$

The state transition probability and the measurement probability together represent the dynamical stochastic system of the robot and its environment. The evolution of states and measurements defined by the probabilities in (2.1) and (2.2) can be described as the *Dynamic Bayesian Network* (DBN) depicted in Figure 2.1.

Not all variables of the environment can be directly measured by the robot sensors, so it must infer the states which cannot be sensed, maintaining an internal *belief* of them. In probabilistic robotics, a belief distribution assigns a density value to each possible hypothesis with regards to the true state. Belief distributions are posterior probabilities over state variables conditioned on the available data. The belief over the state $x_t$ conditioned on all measurements and control actions are calculated after incorporating the most recent measurement $z_t$, and is defined as:

$$bel(x_t) = p(x_t \mid z_{1:t}, u_{1:t})\,. \tag{2.3}$$

[1]It is assumed in this text that the robot executes a control action and then takes measurements

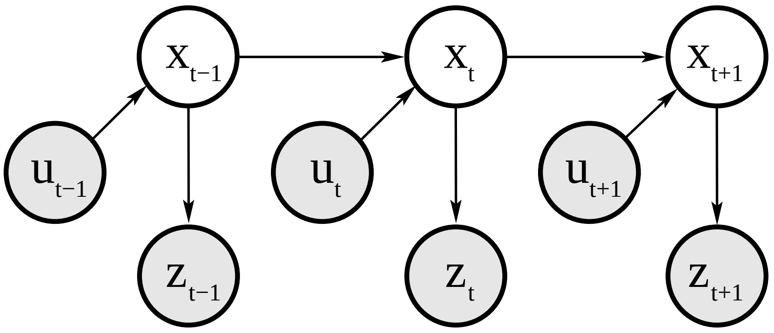


Figure 2.1: Evolution of states and measurements in a DBN. Thrun et al. (2005)

It is also useful to compute the posterior probability of $x_t$ before incorporating the most recent measurement $z_t$, but just after executing the control action $u_t$. This belief is often referred to as the *prediction* or *control update* in Bayesian filters, and is defined as:

$$\overline{bel}(x_t) = p(x_t \mid z_{1:t-1}, u_{1:t}) \,. \tag{2.4}$$

Computing the belief $bel(x_t)$ from $\overline{bel}(x_t)$ is usually referred to as the *correction step* or the *measurement update*.

## 2.3 Bayesian Filters

The most common solution to calculate beliefs from control actions and measurements is the *Bayes filter* algorithm, based on the *Bayes rule*:

$$p(x \mid y) = \frac{p(y \mid x)p(x)}{p(y)} = \frac{p(y \mid x)p(x)}{\int_{x'} p(y \mid x')p(x')dx'} \,,$$

where $p(x)$ is called the *prior probability distribution*, $y$ is the measurement data, and $p(x \mid y)$ is referred to as the *posterior probability* distribution over $x$. Bayes rule provides a convenient way to calculate the posterior $p(x \mid y)$ from its "inverse" conditional probability $p(y \mid x)$ and the prior $p(x)$. It is worth noting that $p(y)$ does not depend on $x$ and can be understood as a *normalization constant* $\eta$, as the sum of the posterior probabilities $p(x \mid y)$ over all the states $x$ must be 1 ($\sum_x p(x \mid y) = 1$). Therefore, Bayes rule can be rewritten as:

$$p(x \mid y) = \eta p(y \mid x)p(x) \,,$$

where $\eta = p(y)^{-1} = (\int_{x'} p(y \mid x')p(x')dx')^{-1}$.

The Bayes filter algorithm pseudocode is stated in Algorithm 1. It is a recursive algorithm that computes the posterior belief $bel(x_t)$ at a time $t$ from the prior belief

$bel(x_{t-1})$ and the most recent control action $u_t$ and measurement $z_t$.

**Algorithm 1** Bayes Filter. Source: Thrun et al. (2005)

```
Require: $bel(x_{t-1})$, $u_t$, $z_t$
Ensure: $bel(x_t)$
1: for all $x_t$ do
2:     $\overline{bel}(x_t) = \int_{x_{t-1}} p(x_t \mid x_{t-1}, u_t) bel(x_{t-1}) dx_{t-1}$
3:     $bel(x_t) = \eta p(z_t \mid x_t) \overline{bel}(x_t)$
4: end for
5: return $bel(x_t)$
```

The Bayes filter algorithm is composed of two essential steps: the *prediction* and the *correction*. The prediction step is done in line 2 by a *convolution* of the prior belief over state $x_{t-1}$ with the state transition probability induced by the control action $u_t$. The correction step, or measurement update, is performed in line 3 by a *product multiplication* of the prediction $\overline{bel}(x_t)$ with the measurement probability $p(z_t \mid x_t)$ of having observed $z_t$ given $x_t$. This procedure is done for all the states $x_t$.

As already mentioned, to return a valid probability distribution, a normalization constant $\eta$ is calculated by summing $bel(x_t)$ for all $x_t$. Then, the calculated beliefs $bel(x_t)$ are normalized by $\eta$, as stated in line 3 of the algorithm.

To compute the posterior belief recursively, an initialization $bel(x_0)$ of the belief is required. If the state $x_0$ is exactly known, $bel(x_0)$ should be initialized as a *Dirac delta function* centered in $x_0$. If the initial state is partially known, $bel(x_0)$ may be represented as a *Gaussian* with mean $x_0$ and variance given by how uncertain is this knowledge. In the extreme case, where the initial state is completely unknown, $bel(x_0)$ should be initialized as an *uniform distribution* over the domain of $x_0$.

### 2.3.1 Mathematical Derivation (Thrun et al. 2005)

The Bayes filter algorithm can be proved by induction. To do so, it must be shown that it correctly calculates the posterior probability distribution $p(x_t \mid z_{1:t}, u_{1:t})$ from the posterior one time step earlier, $p(x_{t-1} \mid z_{1:t-1}, u_{1:t-1})$. The correctness of the proof follows then by induction given that the prior belief at time $t = 0$, $bel(x_0)$ is correctly initialized. Applying Bayes rule to the target posterior yields:

$$\begin{aligned} p(x_t \mid z_{1:t}, u_{1:t}) &\overset{\text{Bayes}}{=} \frac{p(z_t \mid x_t, z_{1:t-1}, u_{1:t}) p(x_t \mid z_{1:t-1}, u_{1:t})}{p(z_t \mid z_{1:t-1}, u_{1:t})} \\ &= \eta p(z_t \mid x_t, z_{1:t-1}, u_{1:t}) p(x_t \mid z_{1:t-1}, u_{1:t}) \end{aligned} \tag{2.5}$$

Equation (2.5) can be simplified with the assumption of state completeness (or the Markov assumption), where if a state $x_t$ is given and a measurement $z_t$ has to be predicted, no past measurements or control actions would provide additional information. This is expressed by the following conditional independence:

$$p(z_t \mid x_t, z_{1:t-1}, u_{1:t}) \overset{\text{Markov}}{=} p(z_t \mid x_t)\,. \tag{2.6}$$

Replacing (2.6) in (2.5) gives:

$$p(x_t \mid z_{1:t}, u_{1:t}) = \eta p(z_t \mid x_t) p(x_t \mid z_{1:t-1}, u_{1:t})\,, \tag{2.7}$$

which is rewritten with the previously defined beliefs:

$$bel(x_t) = \eta p(z_t \mid x_t)\overline{bel}(x_t) \tag{2.8}$$

The next step is to expand $\overline{bel}(x_t)$ using the *Theorem of Total Probability* [2] and simplifying $p(x_t \mid x_{t-1}, z_{1:t-1}, u_{1:t})$ using the Markov assumption again:

$$\begin{aligned}
\overline{bel}(x_t) &= p(x_t \mid z_{1:t-1}, u_{1:t}) \\
&= \int_{x_{t-1}} p(x_t \mid x_{t-1}, z_{1:t-1}, u_{1:t}) p(x_{t-1} \mid z_{1:t-1}, u_{1:t}) dx_{t-1} \\
&\overset{\text{Markov}}{=} \int_{x_{t-1}} p(x_t \mid x_{t-1}, u_t) p(x_{t-1} \mid z_{1:t-1}, u_{1:t-1}) dx_{t-1} \\
&= \int_{x_{t-1}} p(x_t \mid x_{t-1}, u_t) bel(x_{t-1}) dx_{t-1}
\end{aligned} \tag{2.9}$$

Equations (2.9) and (2.8) coincide with lines 2 and 3 in Algorithm 1, respectively.

It is worth noting that some simplifications were done in this derivation regarding the Markov assumptions. In practice, state completeness is hard to achieve, and unmodeled dynamics, inaccuracies in the probabilistic models, and approximation errors are examples of real-world applications that induce violations of the Markov assumption. However, incomplete state representations are usually applied to reduce the computational complexity of the algorithm and Bayes filters have shown to be robust to such violations, obtaining good results with the Markov assumption even for incomplete states. (Thrun et al. 2005).

Bayes filter requires the exact calculation of beliefs, state transition probabilities, and measurement probabilities. However, this is only possible for very simple estimation problems, where these distributions can be represented in closed form. Additionally, the computational complexity of the algorithm turns it impracticable for real-world applications. Therefore, different techniques, based on the Bayes filter, use assumptions and approximations to represent these probability distributions and derive computationally feasible algorithms.

A complete review of several of these algorithms and their applications is found

[2]Theorem of Total Probability: $p(x) = \int_y p(x \mid y)p(y)dy$

in Chen (2003). The most common ones are presented in the following sections.

## 2.4 Gaussian Filters

One way to represent probability distributions is by *Normal Distributions*, which are *Gaussian Functions* that have unitary area. The *Probability Density Function* (PDF) of a unique state normal distribution is parameterized by its mean $\mu$ and covariance $\sigma^2$:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right)$$

Normal distributions over a state vector $x \in \mathbb{R}^n$ are called *Multivariate Normal Distributions*, which are given by:

$$p(x) = \frac{1}{\sqrt{det(2\pi\Sigma)}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) ,$$

where $\mu$ is the mean vector and $\Sigma$ is the covariance matrix, which is a symmetrical positive semidefinite matrix. Normal distributions $p(x)$ are abbreviated as $\mathcal{N}(x;\mu,\sigma^2)$ for the scalar state $x$ or $\mathcal{N}(x;\mu,\Sigma)$ for the multivariate case.

The Gaussian representation of a probability distribution is characterized by two parameters: the mean $\mu$, which is the expected value of the state $x$ and the covariance $\Sigma$, which represents the uncertainty of the state variables in $x$, which may be correlated or not. This parametrization is called the *moments parametrization*, as the mean and covariance are, respectively, the first and second moments of a probability distribution.

The representation of the posterior by a Gaussian has the important characteristic of being unimodal, that is, having only a single maximum value. The implications are that they are suited to tracking problems in robotics, where the posterior is concentrated around the true state with some margin of uncertainty. However, for global localization problems, where multiple hypothesis may be considered, a Gaussian representation may not be a good choice.

### 2.4.1 The Kalman Filter

The *Kalman Filter* (KF) (Kalman 1960) is perhaps the most consolidated technique for state estimation problems. It is an optimal state estimator in the sense that it processes model predictions and measurement corrections to infer a state estimation with minimum error, given that the following conditions are satisfied:

1. The state is continuous. So, Kalman filters cannot be applied to discrete or hybrid state spaces.

2. The state transition probability $p(x_t \mid x_{t-1}, u_t)$ and the measurement probability $p(z_t \mid x_t)$ are linear functions in their arguments with added Gaussian noise.

3. The initial belief $bel(x_0)$ is normally distributed.

These assumptions, together with the Markov assumption in the Bayes filter, ensure that the posterior density $bel(x_t)$ at a time $t$ is always Gaussian in a KF. To fulfill these requirements, the Kalman filter represents posteriors using the moments parametrization. It recursively computes the belief $bel(x_t)$ at time $t$, represented by a mean $\mu_t$ and a covariance $\Sigma_t$, from the belief $bel(x_{t-1})$ at time $t-1$, parameterized by $\mu_{t-1}$ and $\Sigma_{t-1}$. To do so, the state transition probability $p(x_t \mid x_{t-1}, u_t)$ is derived from the linear function:

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t \,, \tag{2.10}$$

where $\varepsilon_t$ is a Gaussian noise (or white noise) vector, which has zero mean and a covariance denoted by $R_t$. This variable models the uncertainty introduced in the system by the state transition.

$A_t$ is a square $n$x$n$ matrix, where $n$ is the number of states, and linearly calculates the state transition. $B_t$ is of size $n$x$m$ with $m$ being the dimensionality of $u_t$, and computes the influence of the control action on the states.

The state transition probability is then given by a Gaussian with mean $A_t x_{t-1} + B_t u_t$ and covariance $R_t$, that is, $p(x_t \mid x_{t-1}, u_t) \sim \mathcal{N}(x_t; A_t x_{t-1} + B_t u_t, R_t)$, or:

$$p(x_t \mid x_{t-1}, u_t) = \frac{1}{\sqrt{det(2\pi R_t)}} \exp\{-\frac{1}{2}(x_t - A_t x_{t-1} - B_t u_t)^T R_t^{-1}(x_t - A_t x_{t-1} - B_t u_t)\} \tag{2.11}$$

Additionally, the measurement probability $p(z_t \mid x_t)$ also has to be *linear Gaussian*, which is defined to be a linear function in its arguments with added Gaussian noise, as is (2.10). The linear Gaussian measurement function is given by:

$$z_t = C_t x_t + \delta_t \,, \tag{2.12}$$

where $C_t$ is a $k$x$n$ matrix that maps states to observations, with $k$ being the number of measurements. The variable $\delta_t$ represents the measurement noise by a multivariate Gaussian with zero mean and covariance $Q_t$. The measurement probability is then given by an MVN with mean $C_t x_t$ and covariance $Q_t$, that is, $p(z_t \mid x_t) \sim \mathcal{N}(z_t; C_t x_t, Q_t)$, or:

$$p(z_t \mid x_t) = \frac{1}{\sqrt{det(2\pi Q_t)}} \exp\{-\frac{1}{2}(z_t - C_t x_t)^T Q_t^{-1}(z_t - C_t x_t)\} \tag{2.13}$$

Finally, the initial belief $bel(x_0)$ is normally distributed with defined mean $\mu_0$ and covariance $\Sigma_0$, that is, $bel(x_0) \sim \mathcal{N}(x_0; \mu_0, \Sigma_0)$, or:

$$bel(x_0) = p(x_0) = \frac{1}{\sqrt{det(2\pi\Sigma_0)}} \exp\{-\frac{1}{2}(x_0 - \mu_0)^T \Sigma_0^{-1}(x_0 - \mu_0)\} \tag{2.14}$$

The recursive Kalman filter pseudocode is shown in Algorithm 2.

**Algorithm 2** Kalman Filter. Source: Thrun et al. (2005)

```
Require: $\mu_{t-1}$, $\Sigma_{t-1}$, $u_t$, $z_t$
Ensure: $\mu_t$, $\Sigma_t$
1: $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$
2: $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$
3: $K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$
4: $\mu_t = \bar{\mu}_t + K_t(z_t - C_t \bar{\mu}_t)$
5: $\Sigma_t = (I - K_t C_t)\bar{\Sigma}_t$
6: return $\mu_t$, $\Sigma_t$
```

Lines 1 and 2 of the algorithm calculates the predicted belief $\overline{bel}(x_t)$ before the measurement incorporation, which is centered at $\bar{\mu}_t$ and given by the deterministic version of the state transition function (2.10), replacing $x_{t-1}$ for $\mu_{t-1}$. The covariance update $\bar{\Sigma}_t$ considers that it is dependent on previous states through $A_t$ and includes the Gaussian noise uncertainty represented by $R_t$, which increases the state uncertainty. Mathematically speaking, the predicted mean $\bar{\mu}_t$ is derived from the minimum of a function $L_t(x_t)$ related to the prediction belief $\overline{bel}(x_t)$:

$$\overline{bel}(x_t) = \int_{x_{t-1}} \underbrace{p(x_t \mid x_{t-1}, u_t)}_{\sim\mathcal{N}(x_t; A_t x_{t-1} + B_t u_t, R_t)} \underbrace{bel(x_{t-1})}_{\sim\mathcal{N}(x_{t-1}; \mu_{t-1}, \Sigma_{t-1})} dx_{t-1} = \eta exp\{-L_t(x_t)\}$$

The first derivative of $L_t(x_t)$ valued at zero gives the equation of $\bar{\mu}_t$ in line 1 of Algorithm 2, while the second derivative yields the equation of $\bar{\Sigma}_t$ in line 2. Lines 3 to 5 computes the state estimation $bel(x_t)$ through the prediction belief $\overline{bel}(x_t)$ by incorporating the measurement $z_t$. It does so by weighting the measurement incorporation in the estimation through the *Kalman gain* $K_t$. Line 4 manipulates the mean by adding to the predicted value $\bar{\mu}_t$ the deviation between the actual measurement and the predicted measurement according to (2.12), adjusted proportionally to the Kalman gain. The covariance $\Sigma_t$ of the posterior belief is computed in line 5, adjusting the prediction covariance $\bar{\Sigma}_t$ with the measurement information gain.

The three equations of lines 3 to 5 are derived from the minimization of a function $J_t$ related to the posterior belief $bel(x_t)$:

$$bel(x_t) = \eta \underbrace{p(z_t \mid x_t)}_{\sim\mathcal{N}(z_t; C_t x_t, Q_t)} \underbrace{\overline{bel}(x_t)}_{\sim\mathcal{N}(x_t; \bar{\mu}_t, \bar{\Sigma}_t)} = \eta exp\{-J_t(x_t)\}$$

The full demonstration of the Kalman filter algorithm is somewhat extensive and will not be reproduced here, but can be found in Thrun et al. (2005). An illustration of the Kalman filter steps for a scalar state is shown in Figure 2.2.

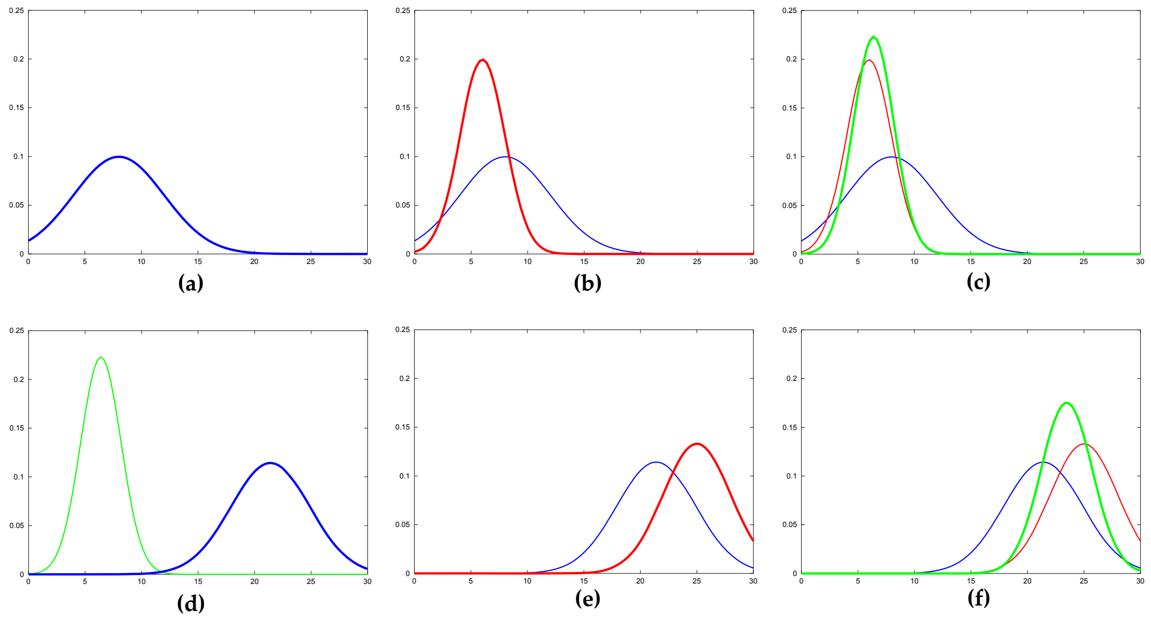


Figure 2.2: Kalman filter steps: (a) initial prediction of the belief; (b) a measurement probability is received, (c) and incorporated to calculate the new belief; (d) the robot motion introduces uncertainty; (e) a new measurement is received; (f) the new belief is computed. Adapted from Thrun et al. (2005)

One can see that a nice feature of the Kalman filter is that the belief variance is lower than the prediction and the measurement variances themselves. Also, the mean of the Gaussian belief is always between the prediction and the measurements means, as the filter is *unbiased*.

Probably, the most important advantage of the Kalman filter is its computational efficiency. The complexity of the algorithm is of the order $\mathcal{O}(k^{2.376})$ (Coppersmith & Winograd 1987), with $k$ being the number of measurements, due to the matrix inversion in line 3, or $\mathcal{O}(n^2)$, with $n$ being the number of states, due to the matrix multiplications in line 5, whichever is the greatest. For comparison, other state estimation algorithms have computational complexity that is exponential in $n$.

On the other hand, as already mentioned, one disadvantage of the KF is its difficulty to deal with multi-hypothesis, as it uses a unimodal representation of the belief. One way to work around this problem is to use a *mixture of Gaussians*:

$$bel(x_t) = \frac{1}{\sum_l \psi_{t,l}} \sum_l \psi_{t,l} det(2\pi\Sigma_{t,l})^{-\frac{1}{2}} exp\{-\frac{1}{2}(x_t - \mu_{t,l})^T \Sigma_{t,l}^{-1}(x_t - \mu_{t,l})\} \,,$$

where $\psi_{t,l}$ are weights of the $l$ mixture components. This type of filter is called *Multi-Hypothesis Kalman Filter* (MHKF).

It is also worth mentioning the *Information Filter* (IF) , which can be understood as the dual of the Kalman filter. It is also a Gaussian filter, but instead of representing Gaussians by its moments, it uses the *canonical parametrization*, which is comprised of an *information vector* $\xi = \Sigma^{-1}\mu$ and an *information matrix* $\Omega = \Sigma^{-1}$.

The duality implies that what is computationally complex in one algorithm is simple in the other, and vice versa. While incorporating measurements is difficult in the Kalman filter due to the matrix inversion in line 3 of Algorithm 2, this is easy in the IF. However, the control update, which is simple in the KF, is involved in the IF. Particularly, the IF is well suited for situations that involve the integration of multiple information, as in multi-robot problems (Fox et al. 2000).

### 2.4.2 Extensions to the Kalman Filter

One strong assumption to the derivation of the Kalman Filter is that the state transition and measurement functions have to be linear. However, in the real world, the majority of systems is nonlinear. If one tries to make a transformation of a Gaussian with a nonlinear function, the result is not a Gaussian anymore, and hence, the KF cannot be applied. Therefore, linearizations have to be done to make the KF practical, considering that the state transition and measurement models are given by the following nonlinear functions:

$$\begin{aligned} x_t &= g(x_{t-1}, u_t) + \varepsilon_t \\ z_t &= h(x_t) + \delta_t \end{aligned}$$

The *Extended Kalman Filter* (EKF) approximates the true belief by a Gaussian performing linearization via *first order Taylor expansion* of the nonlinear models around the most likely state. For the state transition model, this is $\mu_{t-1}$ and for the measurement model, $\bar{\mu}_t$:

$$g(x_{t-1}, u_t) \approx g(\mu_{t-1}, u_t) + \underbrace{g'(\mu_{t-1}, u_t)}_{:=G_t}(x_{t-1} - \mu_{t-1}) \tag{2.15}$$

$$h(x_t) \approx h(\bar{\mu}_t) + \underbrace{h'(\bar{\mu}_t)}_{:=H_t}(x_t - \bar{\mu}_t)\,, \tag{2.16}$$

where $G_t$ and $H_t$ are *Jacobians* and they replace $A_t$ and $C_t$ of the KF algorithm in the EKF. The expected values for the state prediction and the measurement prediction are $g(\mu_{t-1}, u_t)$ and $h(\bar{\mu}_t)$, respectively. Given these linearizations, the EKF algorithm derivation is straightforward, following the same steps of the KF.

Another option of dealing with nonlinearities is the *Unscented Kalman Filter* (UKF) (Julier & Uhlmann 1997). The UKF performs a stochastic linearization using a weighted statistical linear regression process. The algorithm applies the

*Unscented Transform* (UT), which deterministically chooses *sigma points* from the Gaussian mean and covariance, which are passed through the nonlinear functions. Statistical parameters from the resulting set are then computed to obtain a better approximation of the mean and the covariance of the posterior belief.

The UFK linearization through the UT generally provides more accurate results than the first order Taylor expansion linearization in the EKF, yielding better results in the state estimation. A comparison of the EKF and UKF algorithms for state estimation problems is found in Van Der Merwe (2004).

## 2.5 Nonparametric Filters

While Gaussian filters represent posterior beliefs in a fixed functional parameterized form, nonparametric filters approximate the posterior by a finite number of values, each corresponding to a region in the state space. One approach is to discretize the state space in finitely many subregions and represent the cumulative posterior density for each region by a single probability value. This is the case of *Histogram Filters*. Other filters approximate the state space by finitely many random samples drawn out from the posterior distribution, as the *Particle Filter* (PF).

The quality of the approximation depends on the number of subregions or samples used to represent the posterior. When this number goes to infinity, the approximation converges to the true posterior. A great advantage of nonparametric filters is that they are capable of approximating any form of posterior distribution. Particularly, they are well-suited to represent complex multimodal beliefs, and thus are able to solve the global localization and even the kidnapped robot problems, which were defined in Section 1.2.

Moreover, they have the ability to deal with nonlinear transformations of random variables, which the Gaussian filters do not. Finally, nonparametric filters are easy to implement, and have become very popular in recent robotic applications.

The main drawback of nonparametric filters is their computational complexity, which is exponential in the number of states. This may be the number of bins in histogram filters or the number of particles in particle filters, which can be very high (up to hundreds of thousands) to have a good posterior approximation in more complex problems. However, there are techniques to dynamically adapt the number of parameters to represent the posterior belief.

### 2.5.1 The Particle Filter

The particle filter has its basis in *Monte Carlo* methods, due to Metropolis & Ulam (1949), and was introduced in Gordon et al. (1993). Since then, it has found many applications in Bayesian statistics (Smith et al. 2013), where it is commonly known as *Sequential Importance Sampling* (SIS), in computer vision, where it is also referred to as the *Condensation Algorithm* (Isard & Blake 1998), and in artificial intelligence, sometimes under the name of *Survival of the Fittest* (Kanazawa et al. 1995). In the last fifteen years, particle filters have received great attention in robotics as a solution to the robot localization and mapping problem.

The particle filter can be understood as a sampling-based method that uses the *Darwinian* concept of *survival of the fittest*. State samples, denoted *particles*, are drawn from a prior distribution and each particle receives an *importance weight* according to the correctness of the measurements, given the particle state representation. In other words, a particle is a hypothesis of what the true state may be and the importance weight quantifies the correctness of this hypothesis relative to the other ones. After iterations of the method, the fittest particles (the ones with the highest weights) are most likely to survive, while the weakest ones tend to collapse.

This superficial explanation comprises the three basic steps of particle filtering: *sampling*, calculation of *importance factors* (or importance weights, or just weights), and *resampling* (or *importance sampling*). Mathematically speaking, a set $\mathcal{X}_t$ of $N$ particles is defined as:

$$\mathcal{X}_t \coloneqq \{x_t^1, x_t^2, \dots, x_t^N\} \,,$$

where each particle $x_t^n$ is associated to a weight $w_t^n$, forming the set of weights $\mathcal{W}_t$:

$$\mathcal{W}_t \coloneqq \{w_t^1, w_t^2, \dots, w_t^N\}$$

The objective of the particle filter is to approximate the posterior belief $bel(x_t)$ by the set of particles $\mathcal{X}_t$. It does so by incorporating the state hypothesis $x_t$ in the particle set $\mathcal{X}_t$ with probability proportional to its posterior distribution $bel(x_t)$:

$$x_t^n \sim p(x_t \mid z_{1:t}, u_{1:t}) \tag{2.17}$$

As an implication of (2.17), a state space subregion with high density of particles has a higher likelihood of the true state being in there. Just like all the other Bayesian filters, the particle filter calculates the belief $bel(x_t)$ at time $t$ recursively from the belief $bel(x_{t-1})$ one time step earlier. Thus, particle filters construct the particle set $\mathcal{X}_t$ recursively from the set $\mathcal{X}_{t-1}$.

Figure 2.3 helps to understand the particle filter steps. The goal is to approach the *target distribution* $p$, which is $bel(x_t)$, by sampling particles from it. However, it's usually impossible to directly sample from $p$, as $p$ is generally not given in a closed form. Instead, one can sample from a *proposal distribution* $g$, which is commonly chosen to be $\overline{bel}(x_t)$. Thus, the particles are distributed according to the proposal, and do not represent the target distribution properly.

To offset the difference between these two distributions, each particle $x_t^n$ is weighted by the importance factor $w_t^n = p(x_t^n)/g(x_t^n)$. This weight shall be proved in Section 2.5.2 to be proportional to the measurement probability, that is, $w_t^n = \eta p(z_t \mid x_t)$. After the computation of the particle weights, the resampling step chooses particles with probability proportional to their weights to form a new set, usually resulting in denser regions of the posterior representation. The number of particles in the new set is preserved and some particles may have duplicates, as they can be chosen more than one time in the resampling step.

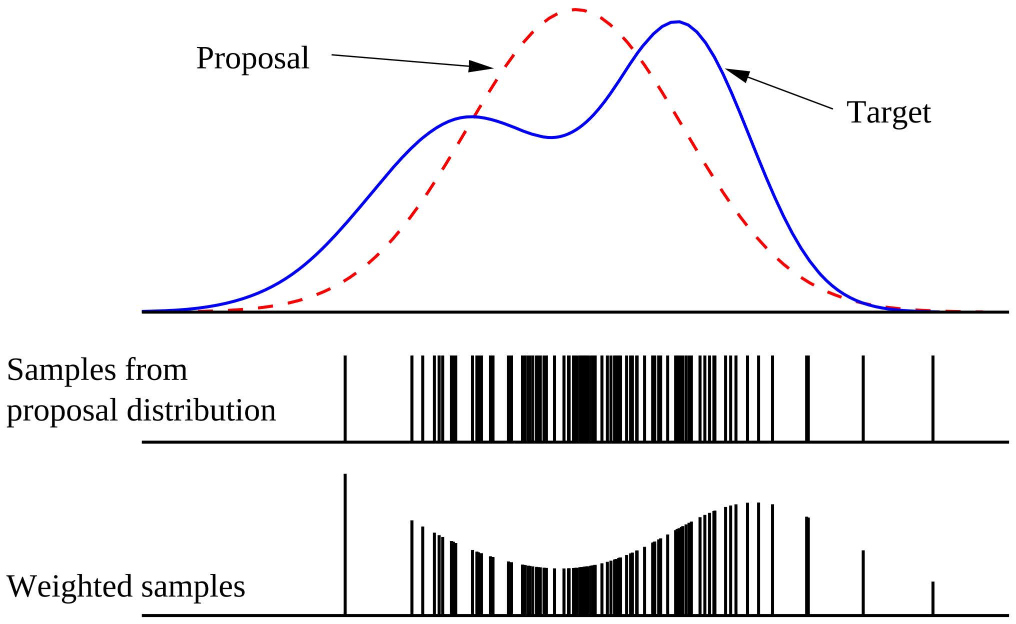


Figure 2.3: Particle filter importance sampling illustration. (Montermerlo 2003)

The standard particle filter algorithm is presented in Algorithm 3. Line 3 samples $N$ particles from the proposal distribution given by the state transition $p(x_t \mid x_{t-1}^n, u_t)$, where $x_{t-1}^n$ are particles from the set $\mathcal{X}_{t-1}$. Line 4 computes the importance factor $w_t^n$ for each particle $x_t^n$ according to $p(z_t \mid x_t^n)$. After sampling $N$ particles and calculating the associated weights, the set $\bar{\mathcal{X}}_t$ represents $\overline{bel}(x_t)$.

Finally, the resampling step is done through lines 7 to 9. The algorithm draws with replacement $N$ particles from the temporary set $\bar{\mathcal{X}}_t$, creating the set $\mathcal{X}_t$. The probability of drawing each particle is proportional to its importance factor. In this process, many particles might have several duplicates. By incorporating the measurements through the weights $w_t$, the new set $\mathcal{X}_t$ is distributed according to the posterior belief $bel(x_t)$.

### 2.5.2 Mathematical Derivation (Thrun et al. 2005)

To derive the mathematical correctness of the particle filter, it is easier to think of particles as samples of state sequences $x_{0:t}^n = \{x_0^n, x_1^n, \ldots, x_t^n\}$. New particles $x_t^n$ are

**Algorithm 3** Standard Particle Filter. Adapted from: Thrun et al. (2005)

```
Require: $\mathcal{X}_{t-1}$, $u_t$, $z_t$
Ensure: $\mathcal{X}_t$
 1: $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
 2: for $n = 1$ to $N$ do
 3:     sample $x_t^n \sim p(x_t \mid x_{t-1}^n, u_t)$
 4:     $w_t^n = p(z_t \mid x_t^n)$
 5:     $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^n, w_t^n \rangle$
 6: end for
 7: for $n = 1$ to $N$ do
 8:     draw $i$ with probability $\propto w_t^i$
 9:     add $x_t^i$ to $\mathcal{X}_t$
10: end for
11: return $\mathcal{X}_t$
```

appended to the sequence of particles $x_{0:t-1}^n$ from which it was generated. With this modification, the posterior belief is calculated over all the state sequences, but does not lose generality:

$$bel(x_{0:t}) = p(x_{0:t} \mid z_{1:t}, u_{1:t})$$

This posterior is expanded analogously to the mathematical derivation of Bayesian filters in Section 2.3.1:

$$\begin{aligned}
p(x_{0:t} \mid z_{1:t}, u_{1:t}) &\overset{\text{Bayes}}{=} \eta p(z_t \mid x_{0:t}, z_{1:t-1}, u_{1:t}) p(x_{0:t} \mid z_{1:t-1}, u_{1:t}) \\
&\overset{\text{Markov}}{=} \eta p(z_t \mid x_t) p(x_{0:t} \mid z_{1:t-1}, u_{1:t}) \\
&= \eta p(z_t \mid x_t) p(x_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t}) p(x_{0:t-1} \mid z_{1:t-1}, u_{1:t}) \\
&\overset{\text{Markov}}{=} \eta p(z_t \mid x_t) p(x_t \mid x_{t-1}, u_t) p(x_{0:t-1} \mid z_{1:t-1}, u_{1:t-1}) \qquad (2.18)
\end{aligned}$$

The derivation is obtained by induction, as in the Bayesian filter derivation. Supposing that the first particle set is obtained by sampling $p(x_0)$ and that the sample set at time $t-1$ is distributed according to $bel(x_{0:t-1})$, the $n^{th}$ particle $x_{0:t-1}^n$ generates the sample $x_t^n$ from the proposal distribution $p(x_t \mid x_{t-1}, u_t) p(x_{0:t-1} \mid z_{1:t-1}, u_{1:t-1})$ with importance factor $w_t^n$, given by:

$$\begin{aligned}
w_t^n &= \frac{\text{target density}}{\text{proposal density}} \\
&= \frac{\eta p(z_t \mid x_t) p(x_t \mid x_{t-1}, u_t) p(x_{0:t-1} \mid z_{1:t-1}, u_{1:t-1})}{p(x_t \mid x_{t-1}, u_t) p(x_{0:t-1} \mid z_{1:t-1}, u_{1:t-1})} \\
&= \eta p(z_t \mid x_t) \qquad (2.19)
\end{aligned}$$

By resampling particles with probability proportional to $w_t^n$, the constant $\eta$ is irrelevant and the resulting particle set is distributed according to the product of the proposal and the weights:

$$bel(x_{0:t}) = \zeta w_t^n p(x_t \mid x_{t-1}, u_t) p(x_{0:t-1} \mid z_{1:t-1}, u_{1:t-1}) \tag{2.20}$$

This proof is only valid for $N \longrightarrow \infty$ due to loss of dimension after normalization. While the non-normalized weights are drawn from an $N$-dimensional space, they reside in $N-1$ dimension after normalization, as the $n^{th}$ weight can be recovered from the $N-1$ other weights by subtracting them from 1. However, this effect is negligible for a high number of particles and the derivation holds true.

### 2.5.3 Discussions about the Particle Filter

In practice, the standard particle filter performs poorly for most applications. One has to understand its properties and practical considerations to make improvements in the algorithm in order to overcome its problems.

#### Sampling Bias

One issue of the particle filter is the *sampling bias*, which is the introduction of bias in the posterior estimate due to the use of finitely many particles. Considering the extreme case of $N = 1$ particle, a single particle will be sampled from the motion model and it will be accepted in the resampling step, regardless of its weight $w_t$, completely ignoring the measurements information. The result is that this particle estimates the density $p(x_t \mid u_{1:t})$ instead of the posterior $p(x_t \mid z_{1:t}, u_{1:t})$.

This is an effect of the normalization step, implicit in resampling, which causes a loss of dimensionality. The use of a sufficient number of particles, usually higher than 100, can turn this issue negligible.

#### Sampling Variance and Resampling

Other important source of error in particle filtering lies on the *variance of the sampler*, which is the variance inherent in random sampling. A finite number of random samples drawn from a probability density has statistics (mean and variance) different from the original distribution from which they were sampled.

The sampling variance can be augmented through repetitive resampling. Considering an extreme case of a static robot that possesses no sensors and is unaware of its state, it should never find out anything about it. Hence, the state estimate at any time $t$ should be exactly the initial estimate.

However, as the standard particle filter outlined in Algorithm  3 performs a resampling step in every algorithm iteration, where some particles are duplicated and others erased, it will occasionally fail to reproduce a particle $x_t^n$, losing particle historical information. As the state transition model in this example is deterministic (the robot is static), no new particles will be generated. The absence of sensing will give equal weights to all particles, and, due to the random nature of resampling, $N$ identical copies of a single particle will survive in the long term.

The resampling process induces a loss of diversity in the particle set. Even though the variance of the particle set itself decreases due to particle replacement, the variance of the estimation of the true belief increases. Therefore, the resampling step must be applied with caution.

**When to Resample?**

A common strategy to alleviate this problem is to reduce the resampling frequency. When the state is known to be static, one should avoid the resampling step (and even the PF iteration at all), as no new information is obtained, and, due to the random nature of the algorithm, particles tend to degenerate. Even when the state is dynamic, resampling should be taken with some criteria, and not on every iteration. To comply with the occasional resampling, the importance weights must keep the historical information of the measurements taken at each iteration of the particle filter. This can be done through the following simple modification:

$$w_t^n = \begin{cases} 1 & \text{if resampling was performed} \\ p(z_t \mid x_t^n) w_{t-1}^n & \text{otherwise} \end{cases} \tag{2.21}$$

If resampling is taken too often, the particle diversity might decrease, while infrequently resampling wastes computational resources, as many particles may be in low probability regions of the state space. A great advantage of occasional resampling is the reduction of computational effort.

When to resample is still an open problem in particle filtering, and, in practice, heuristics are used to determine the resampling times. A simple and somewhat sloppy alternative is to deterministically set a resampling frequency, determined by practical experience. Resampling is done on every $k^{th}$ iteration of the algorithm.

A more elaborated way to establish the resampling times is to keep track of a measure of the sample degeneracy, which may be the variance of the set of importance weights $\mathcal{W}_t$. If this variance is too small, the particles have a rich diversity and resampling should not be taken. A high variance implies particle concentration and resampling should be performed. According to Crisan & Obanubi (2012), this method turns out to return random resampling times.

The most popular metric for the sample degeneracy of a set of particles is the so called *effective sample size* (or *effective number of particles*), introduced in Liu

(1996) (see also Doucet et al. (2000), Liu & Chen (1998)). One cannot evaluate the exact effective sample size, but a good estimation is given by:

$$N_{eff} = \frac{1}{\sum_{n=1}^{N}(w_t^n)^2}\,, \tag{2.22}$$

where $N_{eff}$ is the effective sample size, $N$ is the number of particles, and $w_t^n$ are the *normalized* importance weights. When $N_{eff}$ is below a fixed empirical threshold $N_{thres}$, which is usually set as a fraction of the total number of particles, a resampling procedure is taken. The interpretation of the effective sample size is that any inference based on a weighted sample set of size $N$ will be approximately as accurate as one based on an independent sample set with size $N_{eff}$.

**Low Variance Samplers**

A second possible solution to the *particle degeneration problem* is to decrease the variance due to resampling by implementing a *low variance sampler*. Low variance samplers are improved algorithms over *multinomial resampling*, which is the ordinary resampling method of independently drawing $N$ random particles with probability proportional to their weights.

The main three low variance resampling methods found in the literature are the *residual sampling*, the *stratified sampling* and the *systematic sampling*. Briefly speaking, residual sampling is a mostly-deterministic algorithm that enforces the number of duplicates of a particle to be proportional to its weight. Stratified sampling considers subgroups of particles and is performed in two steps. Initially, the number of particles drawn from each subgroup is given by the sum of the weights of the particles contained in it. Then, samples are drawn from each subset using other resampling algorithm. Systematic sampling draws a single random number and systematically cycles through the weight set by adding fixed increments to this random number, picking the particle that corresponds to the resulting value.

Several other resampling techniques are found in the literature, including hybrid methods, as in Bolić et al. (2003), and comparisons of the resampling methods cited above can be found in Douc & Cappé (2005) and Hol et al. (2006).

**The Resampling Wheel**

One example of a systematic resampling algorithm is the *Resampling Wheel* (Thrun 2012), which finds its similarities in genetic algorithms with the *Roulette Wheel Sampling* (Goldberg et al. 1989). The idea is to select samples with a sequential stochastic process instead of selecting particles independently. Rather than choosing $N$ random numbers and selecting samples assigned to these random numbers, a low variance sampler computes a single random number and selects particles according to it, but still proportionally to the particle weights. The resampling wheel algorithm is given in Algorithm 4 and illustrated in Figure 2.4.

**Algorithm 4** Resampling Wheel. Adapted from Thrun (2012)

```
Require: $\bar{\mathcal{X}}_t$, $\mathcal{W}_t$
Ensure: $\mathcal{X}_t$
 1: $\mathcal{X}_t = \emptyset$
 2: $i = \text{rand}(1, N)$
 3: $\beta = 0$
 4: for $n = 1$ to $N$ do
 5:     $\beta = \beta + \text{rand}\{0, 2\max(\mathcal{W}_t)\}$
 6:     while $\beta > w_t^i$ do
 7:         $\beta = \beta - w_t^i$
 8:         $i = i + 1$
 9:     end while
10:     append $\bar{x}_t^i$ to $\mathcal{X}_t$
11: end for
12: return $\mathcal{X}_t$
```

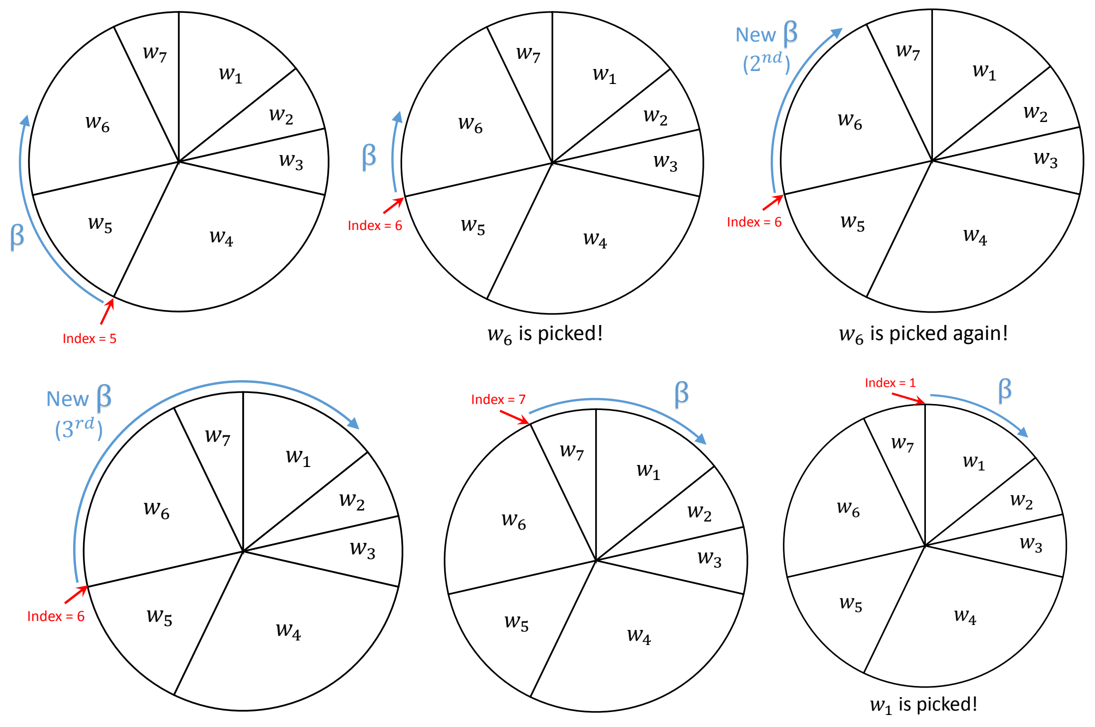


Figure 2.4: Illustration of the steps of the resampling wheel algorithm.

In the resampling wheel algorithm, each particle occupies a slice of a circle with area proportional to its weight. A particle index $i$ is initialized at random in line 2, and a variable $\beta$ is set to zero in line 3. To build the new particle set $\mathcal{X}_t$ from the previous set $\bar{\mathcal{X}}_t$, in the loop in lines 4 to 11, a random value between 0 and

$2\max(\mathcal{W}_t)$[3], which is the largest value in the importance weight set $\mathcal{W}_t$, is added to $\beta$ in each iteration in line 5. The while loop through lines 6 to 9 picks the particle index that corresponds to where the value $\beta$ is sitting on the wheel. Finally, in line 10, the $i^{th}$ particle $\bar{x}_t^i$ in the set $\bar{\mathcal{X}}_t$, is added to the new particle set $\mathcal{X}_t$. In this way, the $\beta$ value cycles throughout the wheel, picking particles with probability proportional to their importance factors.

This low variance sampler has three main advantages over other resampling methods. First, it cycles through all particles in a systematical manner rather then choosing them independently at random. Also, if all particles have the same weight, the resulting set $\mathcal{X}_t$ will be equal to the initial set $\bar{\mathcal{X}}_t$. Finally, its computational effort is lower than other resampling methods, as it requires a complexity of $\mathcal{O}(N)$, while other algorithms require $\mathcal{O}(N \log N)$.

### Particle Deprivation

Another issue to be dealt with in particle filters is the *particle deprivation problem* (or *particle depletion*). In each resampling step, there is a probability higher than zero that particles on the vicinity of the correct state, with high importance factors, are erased, while particles with small weights survive. This happens specially in global localization problems when the estimation is done in a high-dimensional space and the number of particles is insufficient to cover all the relevant regions.

A natural idea to mitigate particle depletion is to increase the number of particles. However, as the computational effort in particle filters increases exponentially with the number of particles, this may be impractical. A good solution is to adapt the number of particles dynamically based on the complexity of the posterior belief. Simple posteriors, which is the case when samples are focused on a single state with small uncertainty, need few particles for a good representation, while complex posteriors, when particles are scattered across the full state domain, demand more samples for a reasonable parametrization. Considering this, in many cases particle filters are implemented as *resource-adaptive* algorithms. They adapt the number of particles based on the available computational resources, as in Kwok et al. (2003).

The *KLD-sampling* algorithm (Fox 2001) is an example of a method that adaptively estimates the number of samples needed on each iteration. It does so by a statistical bound of the particle filter approximation error, which is measured by the *Kullback-Leibler distance* between the PF belief and the true posterior. The algorithm generates particles until this bound is satisfied. The result is that a large number of samples are used if the state uncertainty is high, as in the initial steps of

[3]The value $\max(\mathcal{W}_t)$ is multiplied by 2 here so that the expected value over the random variable $\text{rand}\{0, 2\max(\mathcal{W}_t)\}$ is $\max(\mathcal{W}_t)$. Hence, it is expected to add $\max(\mathcal{W}_t)$ to $\beta$ in each iteration, while keeping the random nature of the algorithm.

a global localization problem, and a small number of particles (typically 1% of the initial number) are used if they are concentrated in a small subregion of the state space. KLD-sampling has shown to consistently outperform fixed sample set sizes in global localization problems by employing a higher number of particles in the initial phase, and much less computational effort in the long run, using less than 10% of the particles used in the fixed sample set approach Fox (2003).

A simpler solution to the particle deprivation problem is to heuristically add a small number of randomly generated particles to the set after each resampling step in order to increase diversity. This method, introduced in Fox et al. (1999), has shown to add robustness to the particle filter and even deal with the kidnapped robot problem. However, it decreases the estimation correctness and should be considered only as a last resource option.

### Divergence between the Proposal and Target Distributions

Another source of error in particle filtering is the divergence between the proposal and target densities. This difference is specially large when the measurement is highly accurate and the motion is not. In this situation, the limited number of samples results in an arbitrarily inefficient particle filter. In other words, the target distribution has narrow peaks, while the proposal is wide. If one uses a small number of samples drawn from the proposal, it may happen that no particles will sit near the narrow peaks of the target density, which will not be correctly represented.

An extreme example is the robot with deterministic sensors, which has the measurement probability $p(z_t \mid x_t)$ as *Dirac delta functions* centered on the states that exactly match the given measurement. Therefore, the proposal distribution will practically never sample particles that exactly correspond to these states, and all importance weights will be zero, turning the filter ill-conditioned.

A close-minded solution in this case is to simply assume more noise in the measurement model than it actually has. Despite being counter intuitive and odd to model a sensor worse than it really is, this can improve the accuracy of some particle filter implementations. Other possible solution is to simply increase the number of particles or use adaptive techniques as the ones described above. However, they can take a long time to converge, and, thus, use a lot of computational power.

A smarter approach is to modify the sampling process and weight computation described in lines 3 and 4 of Algorithm 3. The idea is to sample some particles according to the measurement model, and not the state transition model.

A good alternative to recover from global localization failures that uses this proposal modification is the *Sensor Resetting Localization* (SRL) algorithm, introduced in Lenser & Veloso (2000). In a nutshell, after the sampling, computation of weights, and occasional resampling steps, the algorithm jumps some particles to states that match the most recent measurement if the robot thinks it is lost. Put differently, it samples particles from the measurement model if the non-normalized weights are small, indicating that the robot is lost. The number of particles to be reset is given by how uncertain the robot is about its belief. A threshold value determines wether the algorithm is accounted, and it is used to avoid particle replacement when the robot is confident about its belief. The SRL algorithm is shown in Algorithm 5.

**Algorithm 5** Sensor Resetting Localization. Adapted from Lenser & Veloso (2000)

```
Require: $\mathcal{X}_t$, $\mathcal{W}_t$, $z_t$
Ensure: $\mathcal{X}_t$
1: $N_{new} = (1 - \text{mean}(\mathcal{W}_t)/w_{\text{thres}})N$
2: if $N_{new} > 0$ then
3:     for $i = 1$ to $N_{new}$ do
4:         sample $x_t^i \sim p(z_t \mid x_t)$
5:         Replace sample from $\mathcal{X}_t$ by $x_t^i$
6:     end for
7: end if
8: return $\mathcal{X}_t$
```

In line 1, the number of new samples to be drawn is calculated based on the mean of the non-normalized weights of the set $\mathcal{W}_t$ compared to a threshold $w_{\text{thres}}$. If this value is positive, new samples drawn from the measurement model in line 4 replace samples from the set $\mathcal{X}_t$ in line 5.

The interpretation of the algorithm is that if the average of the particle weights is smaller than an expected value, a fraction of the total particles should be replaced proportionally to the deviation between the average and the threshold. The value $(1-\text{mean}(\mathcal{W}_t)/w_{\text{thres}})$ can be understood as the probability of the robot being wrong about its belief. In the extreme case of $\text{mean}(\mathcal{W}_t) = 0$ (when the robot is completely lost), all particles are replaced, while, if this value is greater than the threshold (the robot has a good idea of where it is), no particle is replaced. The new samples are generated in regions consistent with the most recent sensor readings. As long as the tracking of the true state is working, no new particles are generated from SRL.

Compared to other related particle filter methods, the SRL algorithm generally uses a smaller number of samples, handles larger errors in the model, and deals with global localization and the kidnapped robot problems very well. The drawback of this method is that sampling from $p(z_t \mid x_t)$ is usually hard, unless its inverse possesses a closed form solution.

The *Mixture Monte Carlo Localization* (Mixture MCL) is another algorithm that modifies the proposal distribution in the sampling process (Thrun et al. 2001, 2000). As in SRL, Mixture MCL also samples from the measurement model, but only for a small fixed fraction of all particles (empirically chosen around 10%), and on the very particle filter step, and these particles are merged with the ordinary particle set. Also, the algorithm computes the importance weights of these particles based on the prior distribution according to the following equation:

$$w_t^i = \int_{x_{t-1}} p(x_t^i \mid x_{t-1}, u_t) bel(x_{t-1}) dx_{t-1} \tag{2.23}$$

While this method is mathematically more rigorous than SRL and approximates the true posterior by the dual proposal distribution, it needs to deal with the integral in (2.23) and requires an additional approximation, as $bel(x_{t-1})$ is a set of particles, and not a probability density. The latter issue is usually addressed by *density extraction algorithms*, such as *k-means clustering*, *density trees*, and *kernel density estimation*. As SRL, Mixture MCL yields superior results over the standard particle filter algorithm, and also provides a sound solution to the kidnapped robot problem, adding a level of robustness to system.

## 2.6 Localization

The problem of mobile robot localization is that of estimating the pose of a robot relative to a known map $m$ of the environment. The pose cannot be sensed directly, so it has to be inferred from data. Therefore, the robot motion, provided by the control actions $u$, and the robot perception about the environment, given by the measurements $z$, are integrated through time to estimate the robot pose relative to the map $m$. This procedure is characterized by the Dynamic Bayesian Network of Figure 2.5, where $x$ are the poses to be estimated.

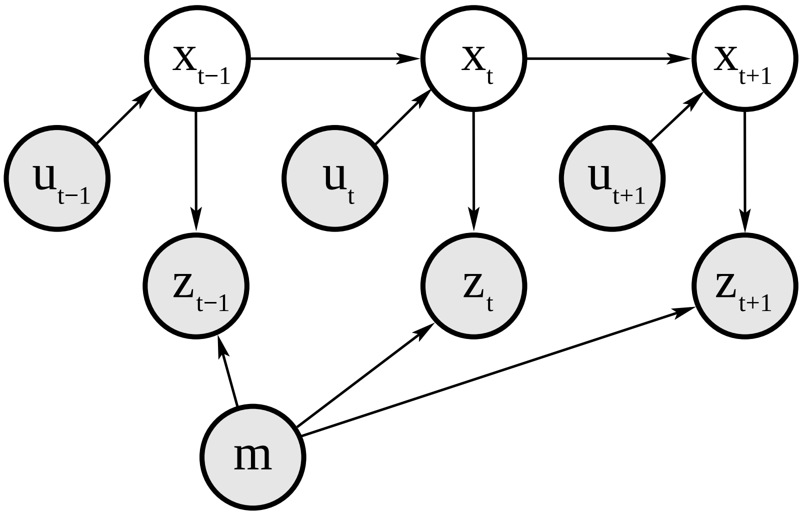


Figure 2.5: DBN of a mobile robot localization. The shaded values are known and the blank cells have to be estimated. (Thrun et al. 2005)

One can see in Figure 2.5 that the map plays an important role in the system, as measurements are conditioned to the map information. Usually, the control actions $u$ are independent from the map, but the poses $x$ are clearly dependent.

Probabilistic localization algorithms are variants of the Bayesian filters presented so far in this chapter with the addition of the map information. Therefore, the state transition and measurement models have to be modified to accommodate the map. Equations (2.1) and (2.2) are respectively redefined as follows:

$$p(x_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t}, m) \overset{\text{Markov}}{=} p(x_t \mid x_{t-1}, u_t, m) \tag{2.24}$$

$$p(z_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t}, m) \overset{\text{Markov}}{=} p(z_t \mid x_t, m) \tag{2.25}$$

### 2.6.1 Markov Localization

*Markov localization* is the straightforward application of the Bayes filter algorithm (Algorithm 1) to the localization problem incorporating the map information, as shown in Algorithm 6.

**Algorithm 6** Markov Localization. Source: Thrun et al. (2005)

```
Require: $bel(x_{t-1})$, $u_t$, $z_t$, $m$
Ensure: $bel(x_t)$
1: for all $x_t$ do
2:     $\overline{bel}(x_t) = \int_{x_{t-1}} p(x_t \mid x_{t-1}, u_t, m) bel(x_{t-1}) dx_{t-1}$
3:     $bel(x_t) = \eta p(z_t \mid x_t, m) \overline{bel}(x_t)$
4: end for
5: return $bel(x_t)$
```

As in Bayes filter, the initial belief $bel(x_0)$ has to be initialized. If the localization problem is one of tracking, then there is some knowledge about the initial pose $x_0$ and $bel(x_0)$ should be a point-mass distribution or a Gaussian centered at $x_0$ with uncertainty given by the covariance $\sigma_0^2$. If the initial pose is completely unknown, $bel(x_0)$ is initialized as a uniform distribution over all the valid poses on the map.

Figure 2.6 illustrates the traditional *one-dimensional hallway robot localization problem* (Thrun et al. 2005). The robot motion model consists on moving to the right, the perception model is given by the probability of identifying a door, and the known map contains three landmarks, which are doors.

Initially, the robot is unaware of its position, and so, $bel(x_0)$ is the uniform distribution represented in Figure 2.6a. The robot does not move in the first iteration ($u_1 = 0$), so $\overline{bel}(x_1) = bel(x_0)$. After sensing a door with the measurement probability density $p(z_1 \mid x_1, m)$, the robot's belief $bel(x_1)$ is updated by multiplying $\overline{bel}(x_1)$ with $p(z_1 \mid x_1, m)$, with implicit normalization, according to line 3 in Algorithm 6 and depicted in Figure 2.6b. At this stage, the poses near the three doors are equally valid hypothesis, and much more probable than the other poses on the map.

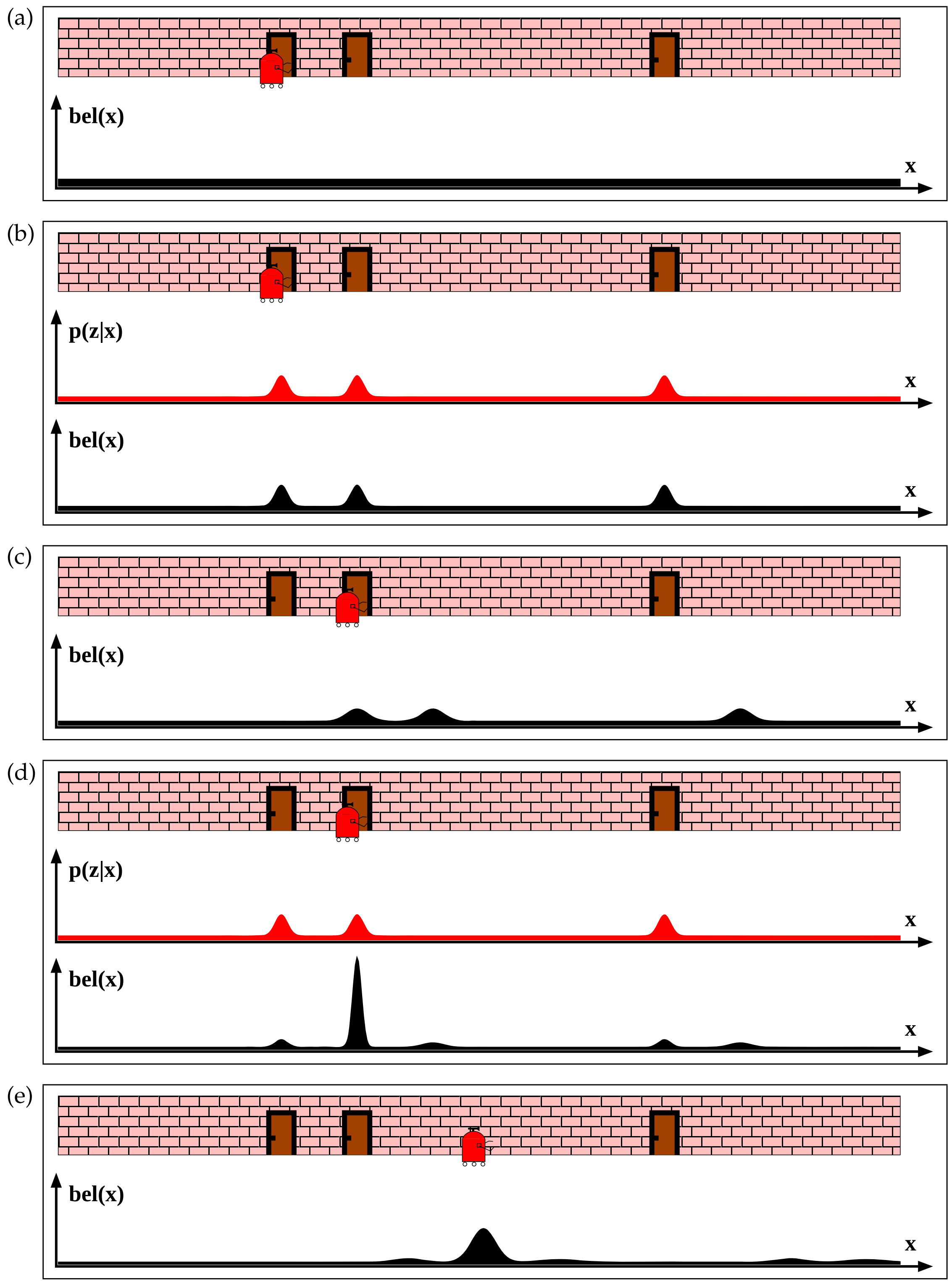


Figure 2.6: Illustration of the Markov localization algorithm. (Thrun et al. 2005)

In a second step, the robot moves to the right with the state transition probability $p(x_2 \mid x_1, u_2, m)$ implicitly given by a Gaussian. The result $\overline{bel}(x_2)$ of the convolution of the previous belief $bel(x_1)$ with the state transition model $p(x_2 \mid x_1, u_2, m)$, given in line 2 of the algorithm, is a flattening on the density, as motion introduces uncertainty (Figure 2.6c). After identifying the second door, the belief $bel(x_2)$ is calculated by the multiplication of $\overline{bel}(x_2)$ and $p(z_2 \mid x_2, m)$ (Figure 2.6d). According to the belief $bel(x_2)$, the robot is already quite confident about its localization. Figure 2.6e shows another motion step and the introduction of uncertainty.

As in Bayes filter, Markov localization needs the state transition and measurement models to be implemented in practice. This can be done using any of the different derivations of Bayes filter already described in this Chapter. In the next two sections, the EKF application to localization is briefly presented, and the particle filter algorithm applied to localization receives more attention.

### 2.6.2 EKF Localization

EKF localization is based on the EKF filter, described in Section 2.4.2, which represents beliefs by Gaussians parameterized by their first and second moments, $\mu_t$ and $\Sigma_t$. The map is seen as a collection of landmarks, that may be uniquely identifiable or not. When the robot cannot distinguish between landmarks in the measurements, it is said that the landmarks have *unknown correspondences* to the measurements. In this case, the algorithm has to estimate these correspondences, as Gaussian filters only deal with single hypothesis due to their natural unimodal characteristic. The EKF localization is described in Algorithm 7, where $G_t$ and $H_t$ are the Jacobians of Equations 2.15 and 2.16.

**Algorithm 7** EKF Localization. Adapted from Thrun et al. (2005)

```
Require: $\mu_{t-1}$, $\Sigma_{t-1}$, $u_t$, $z_t$, $m$
Ensure: $\mu_t$, $\Sigma_t$
 1: $\bar{\mu}_t = g(u_t, \mu_{t-1}, m)$
 2: $\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$
 3: $K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$
 4: $\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t, m))$
 5: $\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$
 6: return $\mu_t$, $\Sigma_t$
```

Figure 2.7 illustrates the one-dimensional hallway example using EKF localization. The landmarks, which are the doors in this example, are assumed to have known correspondences, as the doors are tagged with numbers.

The differences between Figures 2.7 and 2.6 are evident. All the probability densities are represented by Gaussians, which are unimodal representations, the initial belief was supposed to be partially known, and the three doors are distinguishable. Considering these assumptions, the robot did a good tracking of its position.

However, if the initial belief was supposed to be unknown and the landmarks were not uniquely identifiable, the EKF localization algorithm would have poor

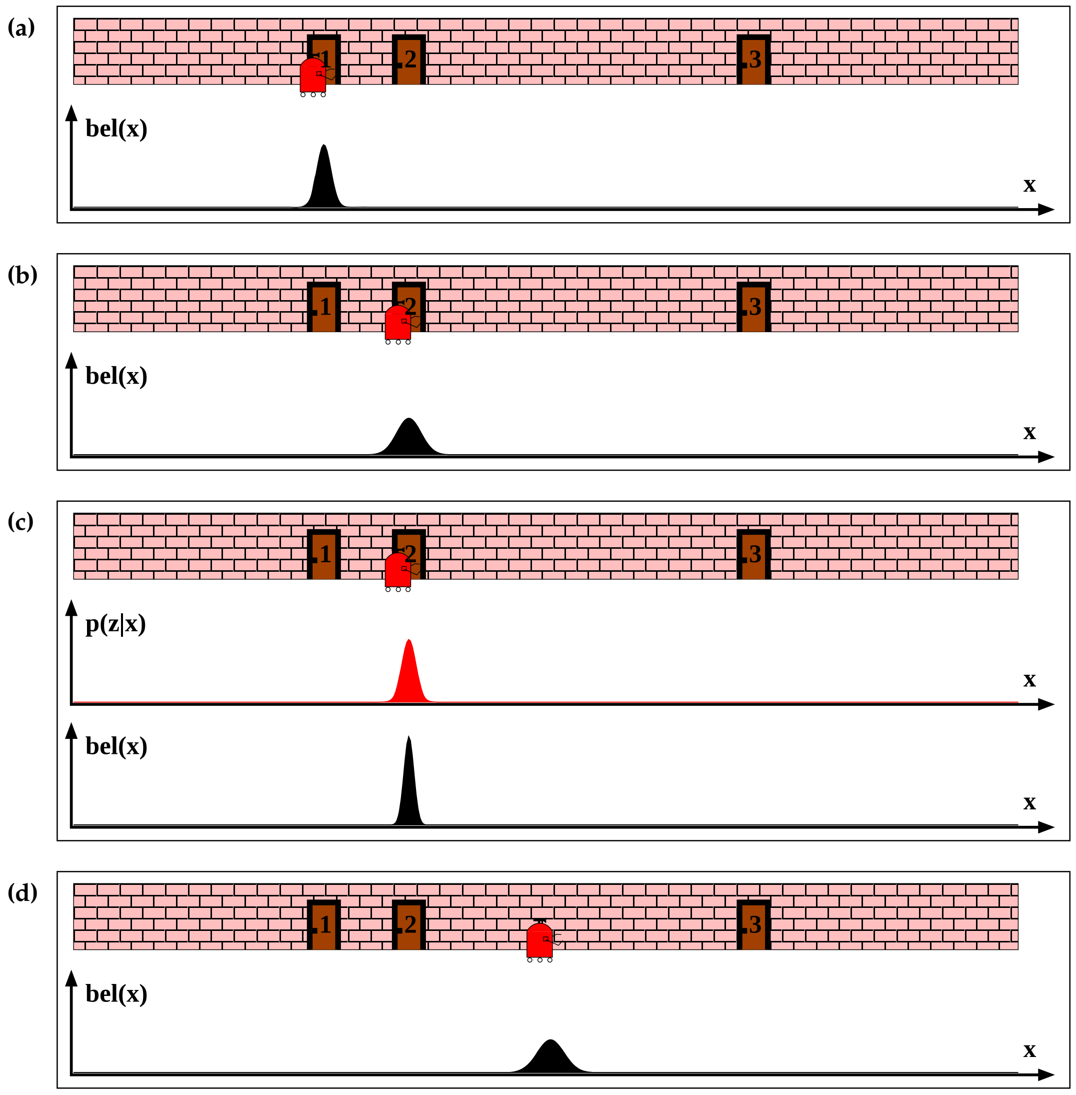


Figure 2.7: EKF localization applied to the hallway example. Thrun et al. (2005)

results, as it does not handle multiple hypothesis well. Variations of the EKF exist in an attempt to deal with this problem, such as *Multi-Hypothesis Tracking* (MHT), already cited in 2.4.1, which maintains multiple EKFs to represent different hypothesis.

### 2.6.3 Monte Carlo Localization

*Monte Carlo Localization* (MCL), due to Fox et al. (1999), is the particle filter algorithm implementation for robot localization problems. It has the following important advantages over the Gaussian localization algorithms:

- MCL can process raw sensor measurements, instead of extracting features from them. As a result, it can also read negative information.
- MCL probability models do not have to be linearized to be represented in a

parametric form, as in EFK localization.

- Due to its non-parametric characteristic, MCL can naturally deal with multiple hypothesis, including unknown landmark correspondences. This implies that it is able to solve global localization and even kidnapped robot problems.

Perhaps the main advantage of MCL is its power to approximate any distribution, not being bounded to a parametric representation, as in EKF localization. However, in many applications, the distributions to be estimated may be intricate, demanding a large number of particles for a reliable representation. Considering that the computational complexity of the algorithm grows exponentially on the number of particles, this is the main drawback of MCL. Nonetheless, this can be overcome by using the methods presented in Section 2.5.3, as SRL. The standard MCL pseudo-code is detailed in Algorithm 8, and illustrated in Figure 2.8.

**Algorithm 8** Monte Carlo Localization algorithm. Source: Thrun et al. (2005)

```
Require: $\mathcal{X}_{t-1}$, $u_t$, $z_t$, $m$
Ensure: $\mathcal{X}_t$
 1: $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
 2: for $n = 1$ to $N$ do
 3:     $x_t^n$ = sample_motion_model($x_{t-1}^n, u_t, m$)
 4:     $w_t^n$ = measurement_model($x_t^n, z_t, m$)
 5:     $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^n, w_t^n \rangle$
 6: end for
 7: for $n = 1$ to $N$ do
 8:     draw $i$ with probability $\propto w_t^i$
 9:     add $x_t^i$ to $\mathcal{X}_t$
10: end for
11: return $\mathcal{X}_t$
```

The modifications of this algorithm relative to the particle filter is the inclusion of the map $m$ in the system models, and the implementation of motion and measurement models to sample from the distribution $p(x_t \mid x_{t-1}^n, u_t, m)$ and compute $p(z_t \mid x_t^n, m)$, respectively. In some cases, the map $m$ plays no role in the motion model, and, thus, is disregarded from the state transition probability.

In Figure 2.8, the robot's belief is initialized as a uniform distribution over all the valid poses on the map. It is considered that the robot does not move in the first step ($u_1 = 0$), so its prior belief $\overline{bel}(x_1)$ is represented by particles sampled from the motion model in line 3 of Algorithm 8 and depicted in Figure 2.8a. After the robot senses a door, the importance weights are calculated in line 4 and associated to the particles in the set $\mathcal{X}_t$ (Figure 2.8b). The robot's prior belief $\overline{bel}(x_2)$ is represented in Figure 2.8c after resampling particles through lines 7 to 10 and sampling the resulting set from the motion model in line 3 when the robot moves to the right.

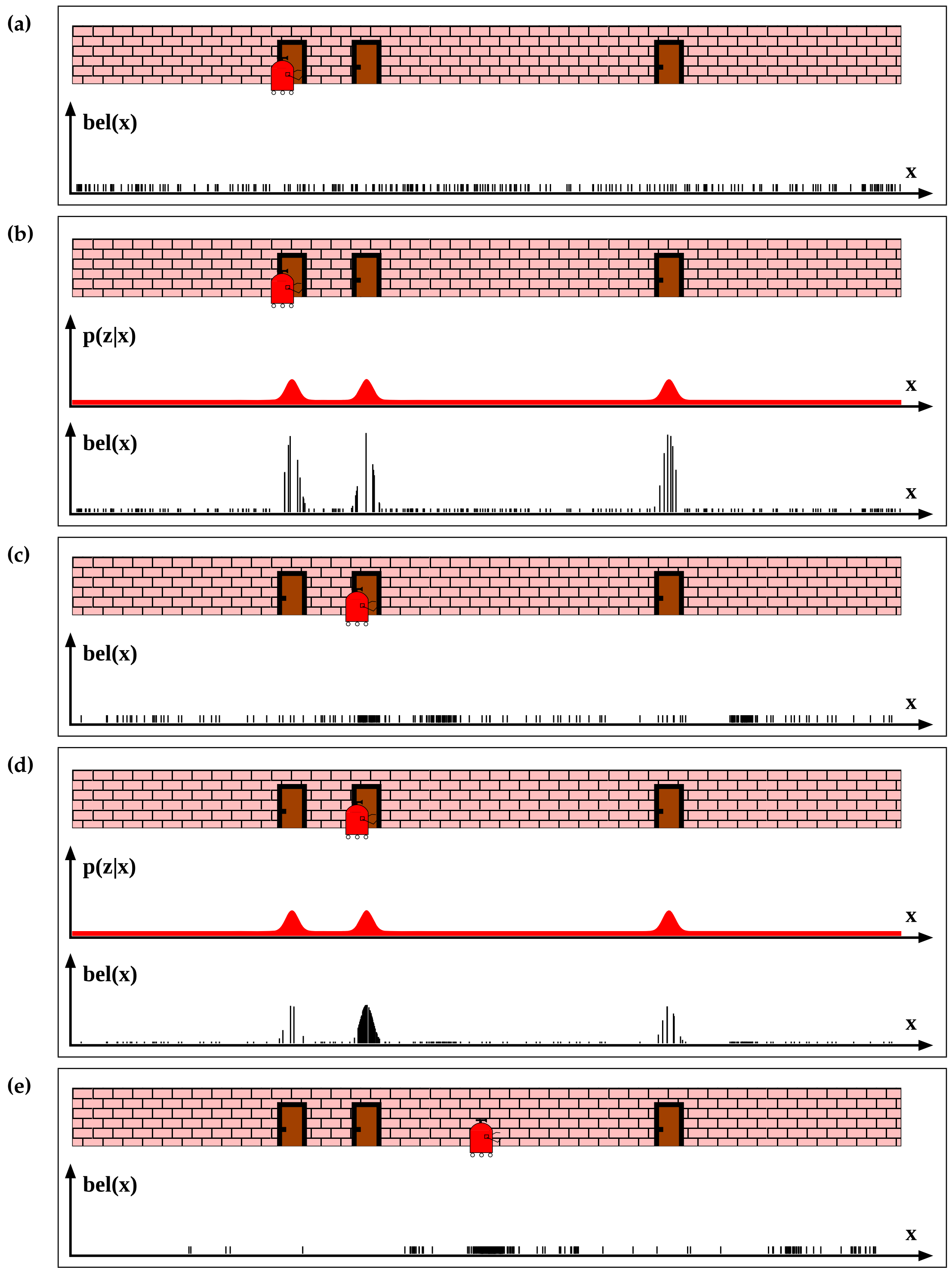


Figure 2.8: MCL applied to the 1-D hallway example. Thrun et al. (2005)

At this time, the robot still keeps three main hypothesis about its position, represented by the high density of particles near them. After sensing the second door and computing the particle weights in Figure 2.8d, the robot has a good estimation of its localization. Figure 2.8e shows the belief after resampling the particle set and moving to the right, when there is a high density of particles near the true position.

# Chapter 3

# DORIS Models

## 3.1 DORIS Environment and Rail Map

At the time of the elaboration of this work, DORIS was being tested on a rail installed in a utility plant at CENPES, a Petrobras research facility, as an intermediate step to the offshore application. The intention was to demonstrate the robot functionalities, mentioned in Section 1.1, in an industrial environment.

The rail path was designed so that the robot could partially fulfill the daily equipment checklist performed by the operators of the utility plant. This includes the visual checking of a diesel generator oil level, switch positions and led status of electric panels, and temperature indicators of boilers.

To comply with the routine inspection characteristic of the daily checklist, the robot periodically travels through the regions of interest on a closed rail path. The closed loop rail also adds a level of safety to the robotic application in the industrial plant, as there is no chance of a derailing due to an eventual unexpected motion caused by a software failure, for example.

Considering this, the rail installed in CENPES has near 130m length and covers almost all the utility plant area of 40m x 20m. It has a base region, with a distance of 1.7m to the floor, where the robot is inserted and removed from the rail by a removable segment. The remaining rail circuit is at a height of at least 2.8m to avoid collision between the robot and passersby. There are occasional level changes on the rail path in order to access regions of interest or deviate from installed equipments.

Any rail used for DORIS may be composed of straight and curved 3in steel tubes. The CENPES rail, in particular, has straight segments, and 45° and 60° curved parts, which all have the same radius of curvature (635mm). The rail is supported by poles fixed to the floor or angle brackets fixed to preexisting beams in the plant.

### 3.1.1 Rail CAD Model

Prior to the rail design, construction and installation, all the $800m^2$ utility plant environment was modelled in a CAD software with a precision of 0.5m. The CAD design is compared to the actual rail installation in Figure 3.1.

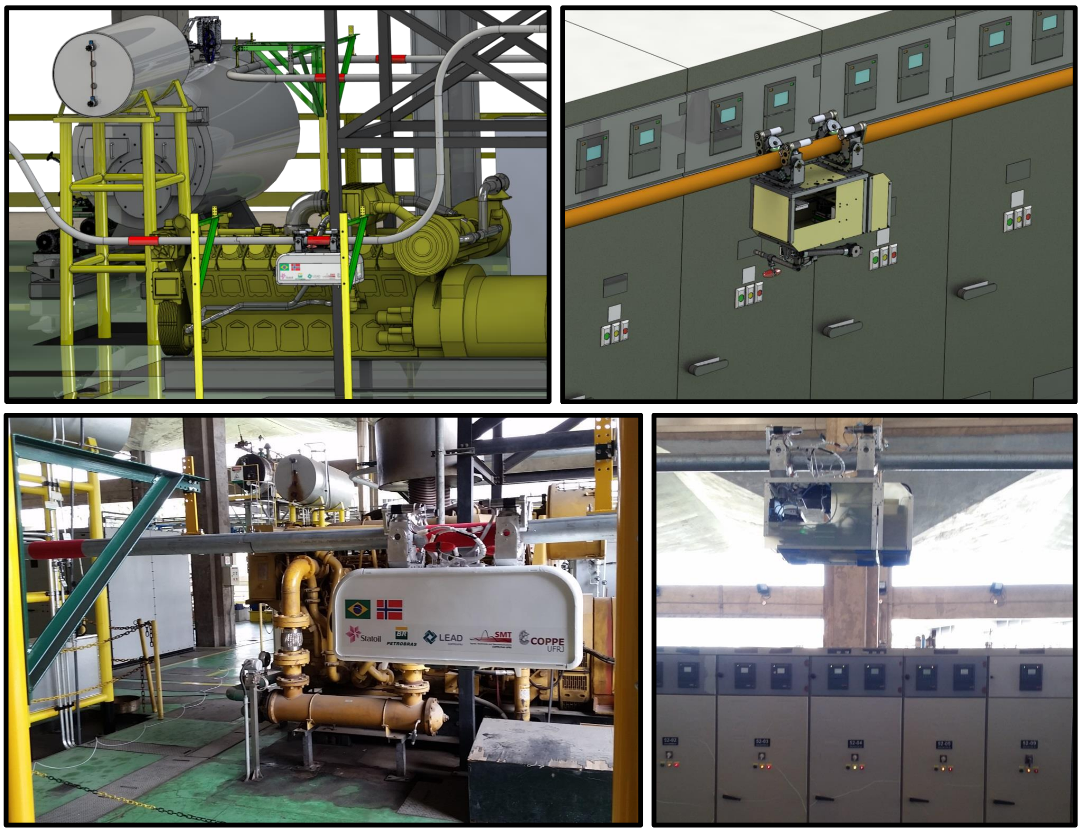


Figure 3.1: CAD design of the rail and the surrounding environment compared to the real world.

After concluding the rail design, fabrication, and installation steps, the CAD model was updated with *as built* measurements of the rail features, as the segments lengths, and the poles positions. However, the CAD model is not completely precise due to the fabrication process and measurement of the rail segments, and to the uncertainty introduced in the rail installation procedure. As an implication of the accumulated uncertainty, the loop does not close in the CAD model when updating it with the *as built* measurements, as shown in Figure 3.2.

The solution to this problem was to estimate the optimal rail path by solving a minimization problem with constraints. The optimization is in the sense of minimizing the errors of the rail segments measurements subject to a loop closure constraint and bounding limits to the length and radius of curvature of the rail segments. One can assume that the rail length designed in CAD corresponds to the real rail length, as the installation and measurements errors are accumulated in the Cartesian space, and not in the rail extension, which is the sum of all the segments lengths. So, this

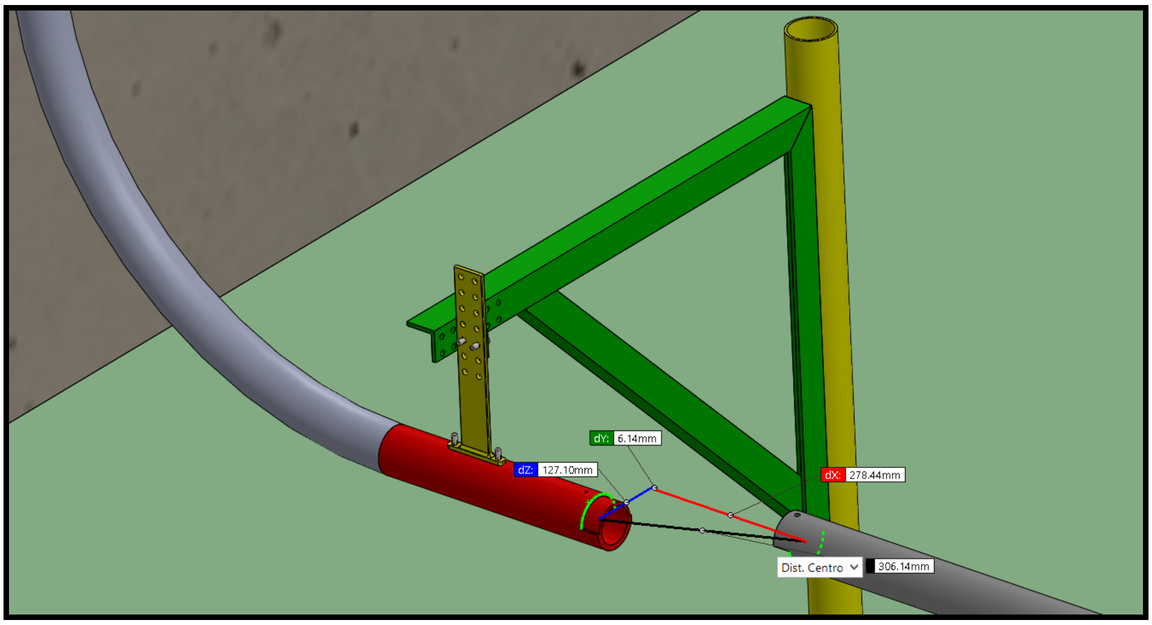


Figure 3.2: The loop of the as built rail does not close in the CAD model.

constraint is also included in the problem.

In order to structure the optimization problem, a straight segment $i$ is parameterized by its length $l_i = \widehat{l_i} + \varepsilon_{l_i}$, where $l_i$ is the real length value, $\widehat{l_i}$ is the measured value, and $\varepsilon_{l_i}$ represents the deviation between the real and measured values. A curved track $j$ is parameterized by a curvature radius $r_j = \widehat{r_j} + \varepsilon_{r_j}$ and an angle $\theta_j = \widehat{\theta_j} + \varepsilon_{\theta_j}$. The total number of segments is $N = N_{\text{straight}} + N_{\text{curve}}$, where $N_{\text{straight}}$ is the number of straight tracks and $N_{\text{curve}}$ is the number of curved tracks. Thus, the total number of parameters of the rail system is $N_p = N_{\text{straight}} + 2N_{\text{curve}}$, as each curve has two parameters.

By properly attaching a coordinate system $E_n$ to each rail part, and knowing the sequence of the installed segments and how the curves are disposed (going up, down, left, or right), one can calculate the homogeneous transformation matrices that represent the rotation and translation between consecutive coordinate frames, which are given by:

$$T_{i,i+1} = \begin{bmatrix} R_{i,i+1} & p_{i,i+1} \\ \mathbf{0}_{1\text{x}3} & 1 \end{bmatrix}.$$

Starting from the first segment and attaching the rail parts sequentially, the loop closure constraint is given by:

$$T_{1,2}T_{2,3}\dots T_{N-1,N}T_{N,1} = T_{1,1} = \begin{bmatrix} \mathbf{I}_{3\text{x}3} & \mathbf{0}_{3\text{x}1} \\ \mathbf{0}_{1\text{x}3} & 1 \end{bmatrix}. \tag{3.1}$$

The loop closure assumption gives 9 equality constraints in the rotation matrix and 3 in the translation vector. All the 12 equations are nonlinear in the parameters $\theta_j$, which turns it a nonlinear optimization problem.

However, given that the fabrication of the curved parts was very accurate ($0.5°$ of tolerance) and that the rail installation was well aligned, one can consider that all the angles $\theta_j$ are known, that is, $\theta_j = \widehat{\theta}_j$. This assumption implies that the rotation matrix of $T_{1,2}T_{2,3}\ldots T_{N-1,N}T_{N,1}$ is the identity, reducing the problem to only 3 linear equations in the translation vector:

$$A\varepsilon = \mathbf{0}\,, \tag{3.2}$$

where $A$ is a 3x$N$ matrix with coefficients calculated from the measured values $\widehat{l}_i$, $\widehat{r}_j$ and $\widehat{\theta}_j$, and $\varepsilon = \{\varepsilon_l, \varepsilon_r\}$ is the $N$x1 vector of errors to be estimated arranged in the correct sequence of the rail path.

There are $2N$ additional constraints by applying lower and upper bounds $lb_n$ and $ub_n$ to each variable $\varepsilon_n$, which limits the measurement and installation errors. Finally, one more constraint is included by making the designed rail length equal to the *as built* rail length, that is, $\sum_{n=1}^{N}\varepsilon_n = 0$.

The objective function must represent the total error to be minimized, so that the sum of squared errors $\varepsilon^T\varepsilon$ is a first choice. However, it is not reasonable to think that the uncertainty in a 10m length section is the same of a 1m track. Therefore, the errors are weighted proportionally to the corresponding segment length by a diagonal matrix $\Lambda$. Representing straight section lengths by $\hat{s}_i = \hat{l}_i$ and curve lengths by $\hat{s}_j = \hat{\theta}_j\hat{r}_j$, the diagonal elements of $\Lambda$ are the inverse of these values arranged in the correct sequence of the rail parts.

The optimization problem is structured below, and it is easily solved by common linear programming algorithms.

$$\begin{aligned}
\underset{\varepsilon}{\text{minimize}} \quad & \varepsilon^T\Lambda\varepsilon \\
\text{subject to} \quad & A\varepsilon = \mathbf{0}\,; \\
& lb_n \le \varepsilon_n \le ub_n, \forall n = 1, \ldots, N\,; \\
& \sum_{n=1}^{N}\varepsilon_n = 0\,.
\end{aligned} \tag{3.3}$$

After solving the optimization problem, the CAD model of the rail is updated and the rail path is sampled with a step of 1cm between consecutive samples (or *track points*). An algorithm removes many samples belonging to straight sections, as they can be represented by only two connected points in the space. Each resulting track point position $[X_s, Y_s, Z_s]^T$ is exported from the CAD model.

### 3.1.2 Rail Frames

The inertial frame of the system is defined with its origin on a known position of the rail base, and corresponds to the first sample point. The $\vec{z}_I$ axis of the inertial frame points up, to the opposite direction of gravity. The $\vec{x}_I$ axis is defined in the along-track direction of the rail, which is positive in the counterclockwise (CCW) direction in $\vec{z}_I$ of the rail loop. The $\vec{y}_I$ axis is obtained from $\vec{y}_I = -\vec{x}_I \times \vec{z}_I$, and coincides with the cross-track direction pointing to the interior of the loop.

To define the position and orientation of each track point $s$ relative to the inertial frame, they also receive coordinate frames $E_s$, which are defined as follows. The $\vec{x}_s$ axis of a track sample $s$ points to the following sample $s+1$, in the along-track direction. The $\vec{y}_s$ axis points to the interior of the rail, in the cross-track direction. The $\vec{z}_s$ axis is obtained from the cross product $\vec{z}_s = \vec{x}_s \times \vec{y}_s$. To characterize the rail path as a loop, the track point that follows the last sample in the set is the first sample point.

The translation between the frames $E_s$ to the inertial frame $E_I$ is given by the track point position $[X_s, Y_s, Z_s]^T$ exported from the CAD model. The rotation matrix can be parameterized by roll ($\psi_s$), pitch ($\phi_s$), and yaw ($\theta_s$) Euler angles for an intuitive representation. However, the *Roll, Pitch and Yaw* (RPY) representation has a singularity in $\phi = \pm\pi/2$, which happens in vertical parts of the rail. In this case, $\psi$ and $\phi$ cannot be uniquely determined, but only their sum $\psi + \phi$.

To solve this singularity, the following important assumption is established:

**A1.** $\psi_s = 0, \forall s = 1, \ldots, N_s$; where $N_s$ is the total number of generated samples.

Despite that the robot has the DoF to roll around the tubular rail, the roll angle is kept to a minimum due to the weight of the robot, and can be neglected in the model. Actually, after several field tests with DORIS, the roll angle measured by an IMU was smaller than 2° in straight sections and 6° in curved and vertical parts, as can be seen in Figure 3.11.

The sampled rail points $s$ are then completely referenced to the inertial frame through the position $[X_s, Y_s, Z_s]^T$ and the RPY Euler angles $[0, \phi_s, \theta_s]^T$. Figure 3.3 shows some generated rail samples in the Cartesian space and coordinate frames represented for a fraction of them.

### 3.1.3 Rail Map

The position of the robot on the rail can be given by a one-dimensional value $s$ that parameterizes the length between the zero point of the rail and the given position in the along-track direction. Based on the definitions given in Section 3.1.2, all the information contained in the rail path can be referenced by the parametric position $s$. A mapping function $f_{map}$ can access this parameter and return the corresponding pose relative to the inertial frame:

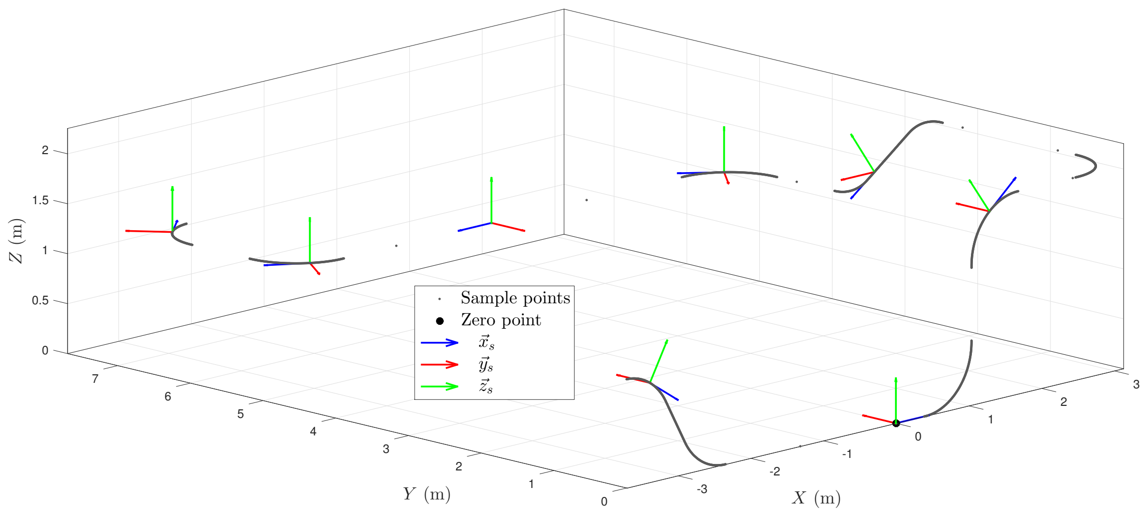


Figure 3.3: Some track points sampled from the rail CAD model and some representative coordinate frames.

$$\begin{bmatrix} X, & Y, & Z, & 0, & \phi, & \theta \end{bmatrix}^T = f_{map}(s, m)\,. \tag{3.4}$$

This continuous function can be built by a *spline interpolation* of the discrete samples poses. A spline interpolation is a continuous piecewise polynomial interpolation that calculates the best set of polynomials that pass through the samples and satisfy some smoothness criteria. In the case of linear polynomials (spline of order one), this function is the *polygonal line approximation*, which interconnects the adjacent points with straight lines.

DORIS rail map is represented by this simplest type of spline, as in RailSLAM (García 2012). The reasons are that 85% of the rail path is composed of straight lines, and that, in curved sections, the resolution of 1 sample per centimeter is good enough to turn the approximation error negligible, which turns out to be 0.2mm in the total length of all curves summed up. For more complex rail paths, an alternative to smoothen the function would be a cubic spline interpolation, as used in Hasberg et al. (2012), where the polynomials are of order 3. However, this adds computational complexity, so that the choice of a linear interpolation for the CENPES rail was sufficient.

Besides of poses on the rail referenced by the parameter $s$, rail features can also be mapped by this value. In the CENPES rail environment, there are natural and artificial landmarks. The natural landmarks are considered the recognizable poles (by a laser scanner) of the rail fixation system, and the artificial landmarks are visual red markers positioned on known rail positions.

Figure 3.4 depicts the rail spline parametrization with the corresponding landmarks positions along with the zero point representation. The "Inside pole" and "Outside pole" are fixation structures that were installed in the interior and exterior regions of the rail loop, respectively. As they can be distinguished in the laser scan, they are represented as different landmarks. The red landmarks, on the other hand, are indistinguishable between them.

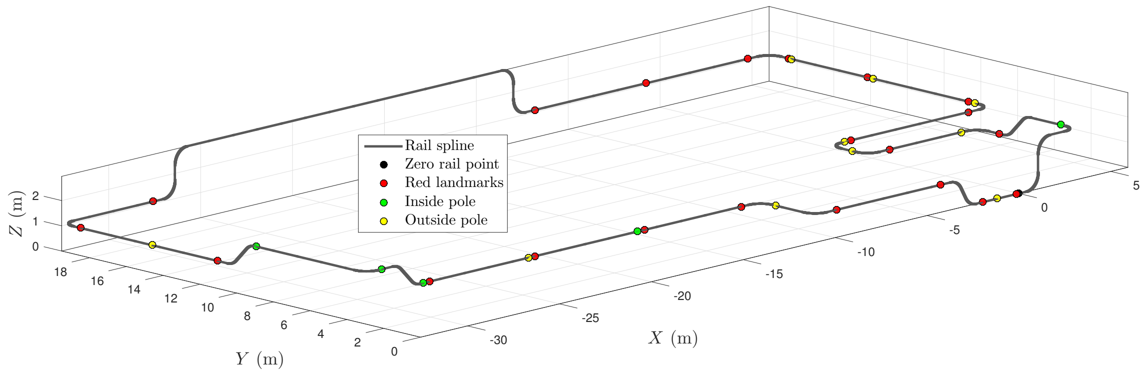


Figure 3.4: Rail spline and landmarks represented in the Cartesian space.

## 3.2 Robot frames

As mentioned in Section 1.2, DORIS moves through the rail suspended by two mechanisms connected to a base (Figure 3.5). Each mechanism has two gimbals (an external gimbal and an internal gimbal) coupled to each other with orthogonal pivot axes, enabling pitch and yaw rotations. The inner gimbal comprises four equally spaced wheels that closely encompass the rail. These mechanisms provide a total of 4 DoF and turn the robot mechanically compliant to any type of rail path.

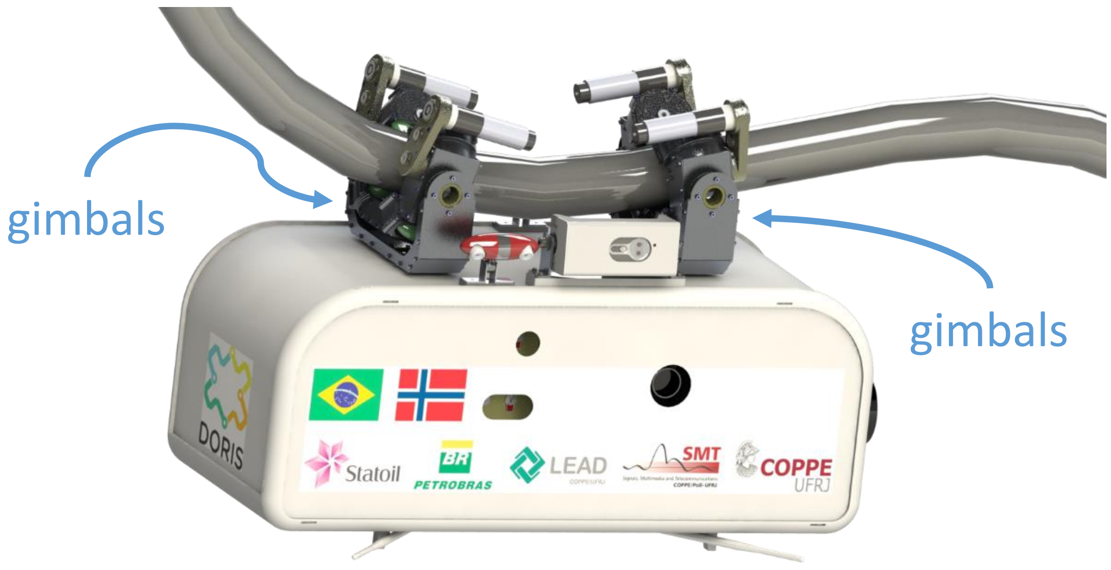


Figure 3.5: DORIS CAD model showing the gimbal mechanisms.

The two gimbal mechanisms are compliant to the rail path, so that it is convenient to represent the coordinate frames $E_1$ and $E_2$ for each of the two gimbals set with origins placed in each intersection point between the internal and external gimbals axes. This point coincides with the rail path, which is defined to be the interior contour of the tubular rail. In this way, the gimbals poses can be obtained through the mapping function $f_{map}(s, m)$ using their parametric values on the rail, $s_1$ and $s_2$, as follows:

$$\begin{bmatrix} X_1, & Y_1, & Z_1, & 0, & \phi_1, & \theta_1 \end{bmatrix}^T = f_{map}(s_1, m)\,, \tag{3.5}$$

$$\begin{bmatrix} X_2, & Y_2, & Z_2, & 0, & \phi_2, & \theta_2 \end{bmatrix}^T = f_{map}(s_2, m)\,. \tag{3.6}$$

The parametric position of the rear gimbal (2) is obtained from the position of the frontal gimbal (1) from $s_2 = s_1 - d$, where $d$ is the constant distance between the gimbal frames $E_1$ and $E_2$. This is actually an approximation, as in curved sections, the arc length between the two gimbal frames is greater than $d$. However, this difference is smaller than 5mm for the considered rail curvature of 635mm, and can be neglected.

It is also useful to attach a coordinate frame to the robot base, which is fixed to the two gimbal mechanisms, as the robot sensors are mounted on it. However, the robot base is not solidary to the rail path when the robot is on a curved section. Therefore, its pose cannot be directly mapped by the rail length parameter using the function $f_{map}(s, m)$. However, it can be inferred from the poses of the frontal and the rear gimbals.

The origin of the base frame is defined as the midpoint of the line that connects the two gimbal frames. The $\vec{x}_b$ axis points in the direction of $E_2$ to $E_1$. To derive the second axis, and, thus, completely define the coordinate frame after calculating the third axis by the cross product between the other two, the following second assumption has to be considered:

**A2.** The robot does not move on two curves at the same time.

Although DORIS is mechanically capable of moving on two or more curves at the same time, the rail installed in CENPES does not have this type of track. This assumption implies that at least one of the $\vec{y}$ and $\vec{z}$ axes of both gimbals set will be coincident, and, thus, this axis will also be the same for the base frame.

DORIS perception system is composed of three sensors, which are detailed in Section 3.4: an IMU, a camera, and a laser scanner. All the three sensors are mounted on the robot base, and, so, they are solidary to the base motion. This implies that the translations and rotations of the sensors frames relative to the base frame are constant.

The IMU is mounted with the $\vec{x}_{IMU}$ axis pointing to the cross-track direction, as is $\vec{y}_b$, and the $\vec{y}_{IMU}$ axis pointing to the along-track direction, as is $\vec{x}_b$. The rotation matrix between the IMU and the base frames is given below. The IMU measurements given in the local frame have to be represented in the robot base frame using the rotation matrix below:

$$R_{b,IMU} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}. \qquad (3.7)$$

As just the orientations and angle velocities measured by the IMU are used in this work, the translational component of the homogeneous transformation between the IMU and the base frames is irrelevant.

Regarding the camera frame, only its position in the along-track direction relative to the base frame is significant in the model. The camera was mounted aligned to the base frame so that this difference is zero.

The laser scanner is mounted on the rear part of DORIS, aligned to the robot symmetrical right plane. The homogeneous transformation $T_{b,\text{laser}}$ that takes a vector represented in the laser frame $E_{\text{laser}}$ to the robot base frame $E_b$ is given as follows:

$$T_{b,\text{laser}} = \begin{bmatrix} 0 & 0 & -1 & -L_x \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & -L_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad (3.8)$$

where $L_x$ and $L_z$ are the distances between the two frames in the $\vec{x}_b$ and $\vec{z}_b$, respectively. Figure 3.6 shows all the relevant robot frames cited in this section.

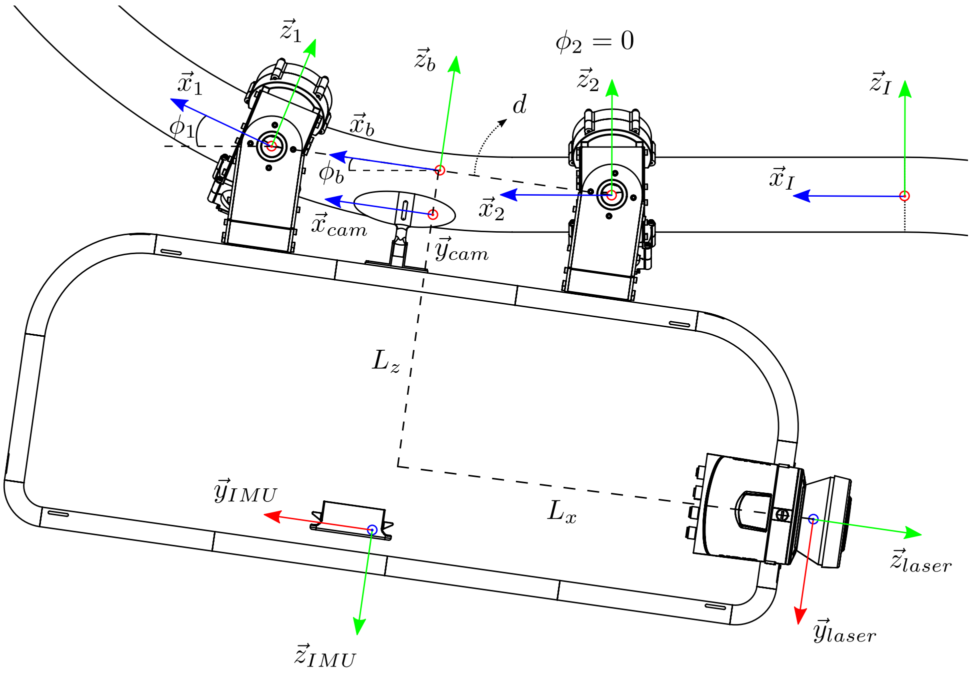


Figure 3.6: DORIS frames: inertial, gimbals, base, IMU, camera, and laser scanner.

The following motion and perception models apply to the specific environment of DORIS rail installed in CENPES. However, the models presented here can be easily generalized to any environment, given that the assumptions A1 and A2 are satisfied, and that the rail map is known.

## 3.3 DORIS Motion Model

After defining the robot main coordinate frames and the transformation functions between them, we can derive the motion and perception models to be used in DORIS probabilistic filter. As we have the transformation matrices between the frames themselves and the inertial frame, which are calculated from a single parameter $s$, the robot systems can be completely represented by this single value. Thus, the frontal gimbal parametric position, $s_1$, is chosen as the variable to be estimated. From now on, $s_1$ will be simply denoted as $s$.

DORIS moves through the rail using a velocity control provided by Maxon EPOS2 70/10 controllers. The velocities of the motors are measured by digital encoders, and converted to the wheels speeds using a relation obtained from the gear ratio of the transmission system, and the wheel diameter. The position $s_t$, at a time $t$, can be calculated from the position $s_{t-1}$ one time step earlier added to the displacement $\Delta s_t = v_t \Delta T$, where $v_t$ is the measured velocity, and $\Delta T$ is the time increment between two state evaluations.

Thus, the state transition probability $p(s_t \mid s_{t-1}, u_t)$ can be given by the Gaussian in (3.10) and can be easily sampled from the linear Gaussian function in (3.9):

$$s_t = s_{t-1} + v_t \Delta T + \varepsilon_t(v_t)\,, \tag{3.9}$$

$$p(s_t \mid s_{t-1}, u_t) \sim \mathcal{N}(s_t; s_{t-1} + v_t \Delta T, \varepsilon_t(v_t))\,, \tag{3.10}$$

where $u_t = v_t \Delta T$ is the control action at time $t$ and $\varepsilon_t$ is a Gaussian noise with standard deviation $\sigma_t^{\text{motion}} = \alpha_{odom} |v_t \Delta T|$, where $\alpha_{odom}$ is the odometry error, given as a percentage of the displacement $v_t \Delta T$. This random noise models the odometry uncertainty, which is reasonable to think as being proportional to the displacement $\Delta s_t = v_t \Delta T$. It is worth noting that the map $m$ was disregarded in (3.10), as it adds no information in the state transition model in the case of DORIS.

## 3.4 DORIS Perception Model

The prediction step of a probabilistic filter, given by the motion model, introduces uncertainty in the estimation. Suppose that the robot is devoid of sensors and start moving through its environment with a known initial condition and an inaccurate motion. The introduction of uncertainty in the motion update step implies that, in steady state, the robot will be completely lost, with a posterior density uniformly distributed over all the possible states, characterizing *maximum confusion*.

Therefore, measurements have to be taken and integrated with the prediction estimate to keep uncertainty to a minimum. DORIS has several embedded sensors that extract useful information from the environment, and some of them can be used for localization purposes. In this work, a camera is used to detect artificial landmarks positioned on the rail, an IMU provides orientation measurements, and a laser scanner extracts localization information from natural features of the rail and the surrounding environment.

### 3.4.1 Camera Measurements

DORIS has a low cost *Red, Green and Blue* (RGB) camera mounted on the robot base and pointing to the rail, which has 20 red sheets manually placed on known positions (Figure 3.4). A particular region of the rail in CENPES is at a height of 4.5m, being unreachable without scaffolds, so that no markers were positioned on it, although they would be useful. The idea is to detect the markers according to the predominance of red pixels on the image frame. As the camera is always framing the rail, which has a naturally grey surface, and the markers are red, the color change is very distinguishable, and a positive detection can be associated to known positions. Figure 3.7 shows DORIS approaching a *red landmark* on the rail base area.

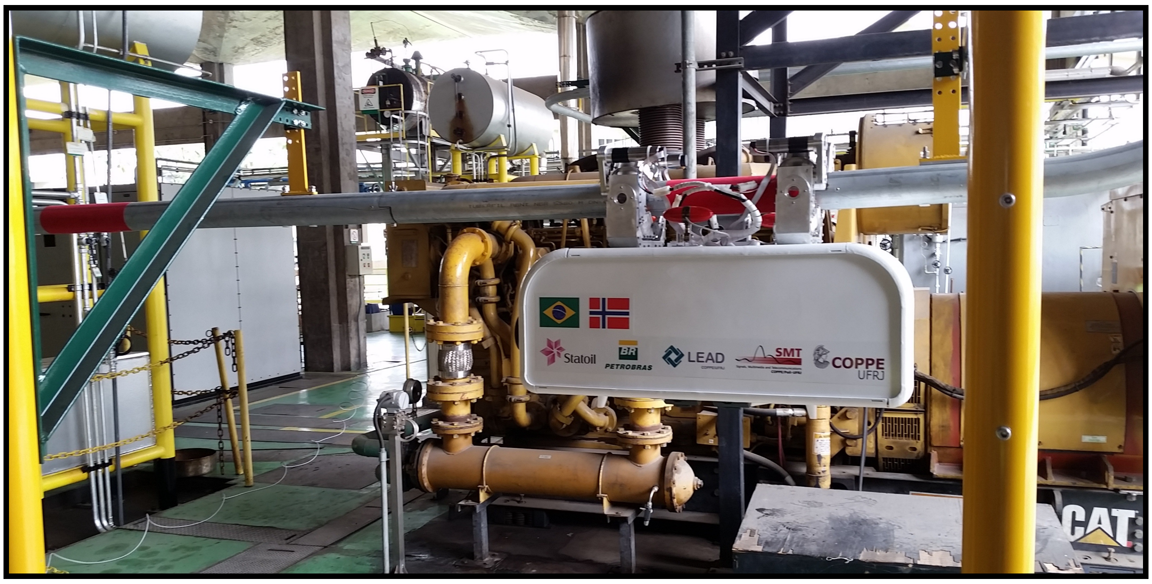


Figure 3.7: DORIS near two red markers installed on the rail base area.

The red detection is done by a simple image processing algorithm using the *OpenCV* library. The image frames are received in RGB values, and are converted to the *Hue, Saturation and Value* (HSV) representation, which is perceptually more relevant than RGB. A filter over the HSV values is applied with minimum and maximum values for each component to distinguish the red pixels on the image. An additional filter is implemented to eliminate noisy pixels, and the result is a boolean matrix of white and black pixels representing if each pixel has the predominance of the red color. The percentage of white pixels over the total number of pixels in the image is used in the localization algorithm. Figure 3.8 shows a camera frame in RGB values and the resulting black and white matrix after processing the image with the *red detection algorithm.*

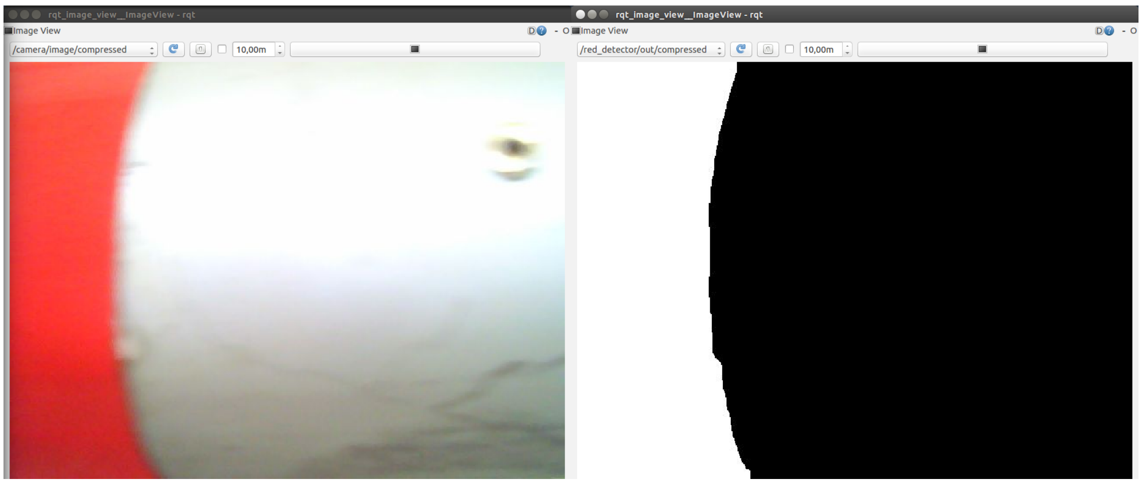

rqt_image_view__ImageView - rqt
Image View
/camera/image/compressed
10,00m
rqt_image_view__ImageView - rqt
Image View
/red_detector/out/compressed
10,00m

Figure 3.8: Original RGB image frame and the resulting black and white image representing the predominance of the red color.

The main advantages of using the visual detection of this type of artificial landmark is that the markers are very simple and easy to include in the environment, and the software implementation is trivial, as described above. The main drawbacks are that the landmarks have unknown correspondences, that is, they are indistinguishable from each other, they can wear over time, and the detection is subject to light conditions.

An alternative approach was tested to eliminate the correspondence problem between the landmarks. QR codes were positioned on known locations of the rail to be detected by the camera through a QR detection algorithm. However, after a few field tests with DORIS, it was verified that the camera image gets significantly blurred while the robot is moving, even with very low speeds (0.1m/s, or 10% of the maximum speed), so that the QR code is not identified by the algorithm. As a result, the simpler colored features were found to be more appropriate.

The detection of a red landmark can be modeled by a boolean variable, where $z_t^{red}$ represents a positive detection and $\neg z_t^{red}$ a negative detection. Based on the known positions of the red markers on the rail, we can derive the probability $p(z_t^{red} \mid s_t, m)$ of detecting a red landmark given a position $s_t$. A positive detection $z_t^{red}$ is given when the percentage of white pixels over the total number of pixels in the camera frame exceeds an empirical threshold, obtained after analysis of several field tests. With this boolean value, the current position $s_t$, and the red markers positions, the probability $p(z_t^{red} \mid s_t, m)$ is modeled by:

$$p(z_t^{red} \mid s_t, m) \sim \mathcal{N}\left(s_t - s_j^{red}; 0, \sigma^{red^2}\right) , \tag{3.11}$$

where $s_j^{red}$ is the closest red landmark position to $s_t$ and $\sigma^{red}$ is the standard deviation used to accommodate small errors in the model, as the landmarks positions and the total lap length uncertainties. Obviously, the probability of not detecting the red landmark on a position $s_t$ is obtained by $p(\neg z_t^{red} \mid s_t, m) = 1 - p(z_t^{red} \mid s_t, m)$. The red detection probability is computed through Algorithm 9.

**Algorithm 9** Red Landmark Detection

```
Require: $s_t$, RGB image frame, $m$
Ensure: $p(z_t^{red} \mid s_t, m)$
 1: Convert image frame from RGB to HSV
 2: Filter image with HSV min. and max. values for red predominance
 3: Filter noisy pixels
 4: Compute the percentage $red_{\%}$ of white pixels over the total number of pixels
 5: Search for the nearest red landmark $s_j^{red}$ to $s_t$
 6: Compute $p(z_t^{red} \mid s_t, m) \sim \mathcal{N}\left(s_t - s_j^{red}; 0, \sigma^{red^2}\right)$
 7: if $red_{\%} < red_{thres}$ then
 8:     $p(z_t^{red} \mid s_t, m) \leftarrow 1 - p(z_t^{red} \mid s_t, m)$
 9: end if
10: return $p(z_t^{red} \mid s_t, m)$
```

Figure 3.9 shows the normalized distribution, in the sense of the maximum value truncated to 1, of a positive red landmark detection over all positions $s$ on the rail using $\sigma^{red} = 0.3$m and a detection threshold $red_{thres} = 75\%$. The model is compared to the real test percentages of white pixels.

According to Figure 3.9, the model is good enough in comparison to the real measurements. However, in this specific test, one of the red landmarks was not detected, due to excessive light in that region. The image taken was so bright that the white color predominated over the red. Note that the phase difference between the model and the test graphs is due to odometric errors, which are associated to the position axis of the test graph.

### 3.4.2 IMU Measurements

DORIS has an embedded *Spatial OEM IMU*, of Advanced Navigation (Figure 3.10). The device comprises three MEMS gyroscopes, three accelerometers, a GNSS receiver, three magnetometers, and other sensors used for calibration purposes.

The drift error of MEMS gyroscopes is typically of $3°/h$ to $20°/h$, but the ac-

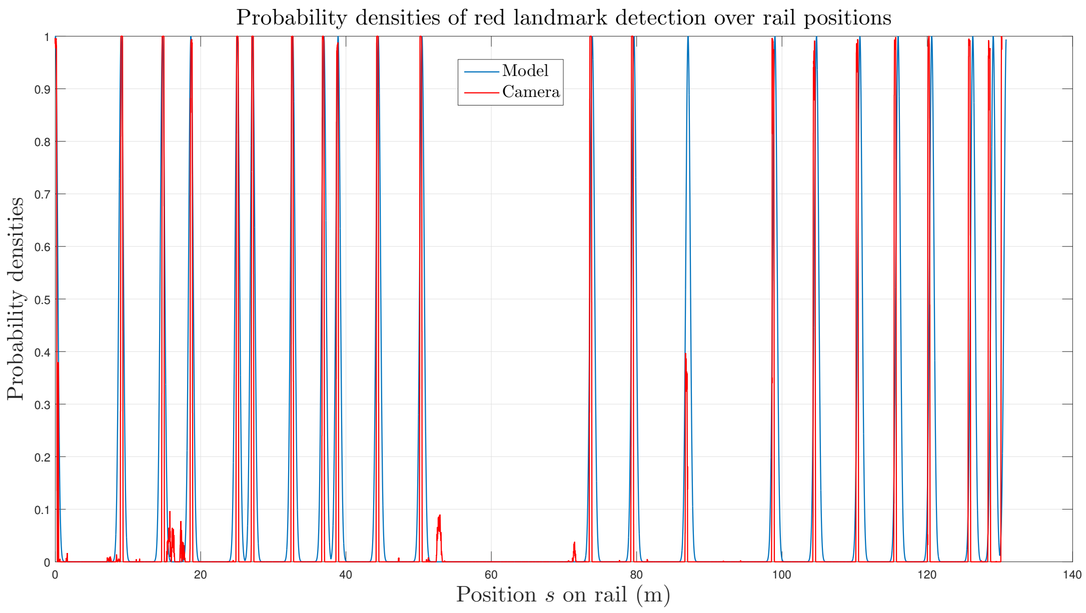


Figure 3.9: Normalized probability density of detecting a red landmark over $s$.

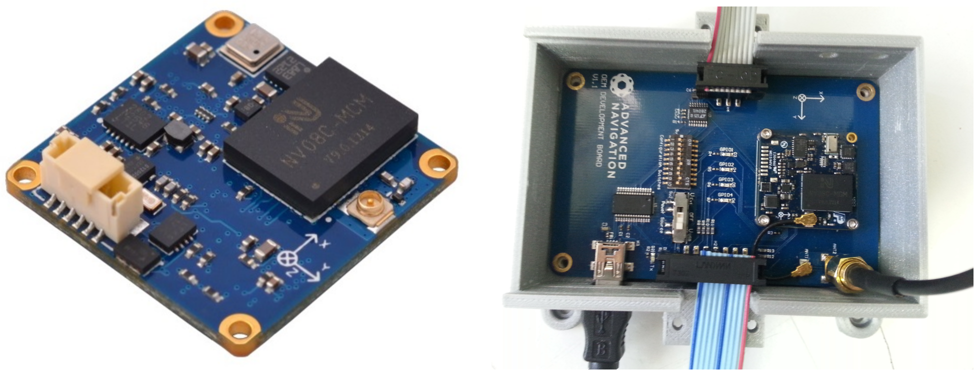


Figure 3.10: Advanced Navigation Spatial OEM inertial measurement unit.

celerometers measurements can limit this error by estimating the roll and pitch angles of the body through gravity information. On the other hand, to infer the body heading (or yaw), an additional measurement is required. Otherwise, the heading angle will be obtained by integrating the gyroscopes measurements, which typically results in drift issues. The common way to derive the heading direction, and hence the yaw angle, is to use magnetometers to search the Earth's North Magnetic Pole, as is the case of the Spatial OEM IMU.

However, the typical measurement accuracy of the heading direction by magnetometers is of $0,5°$ to $10°$, and it can be severely degraded by disturbances on the nearby magnetic field. Static magnetic interferences can be compensated with a previous calibration, but dynamic disturbances are a much bigger issue. Unfortunately, the CENPES utility plant has several ferromagnetic structures and equipments that generates significant static and dynamic disturbances on the magnetic field, which invalidate the yaw values measured by the IMU.

Figure 3.11 compares the RPY values obtained from the model, given by the mapping function in (3.4), with the IMU measurements for positions within a complete rail loop. The device is installed on the robot base (Figure 3.6), so that the IMU measurements are computed in the base frame after taking the readings provided in the IMU frame to the base frame by the static transformation in (3.7).

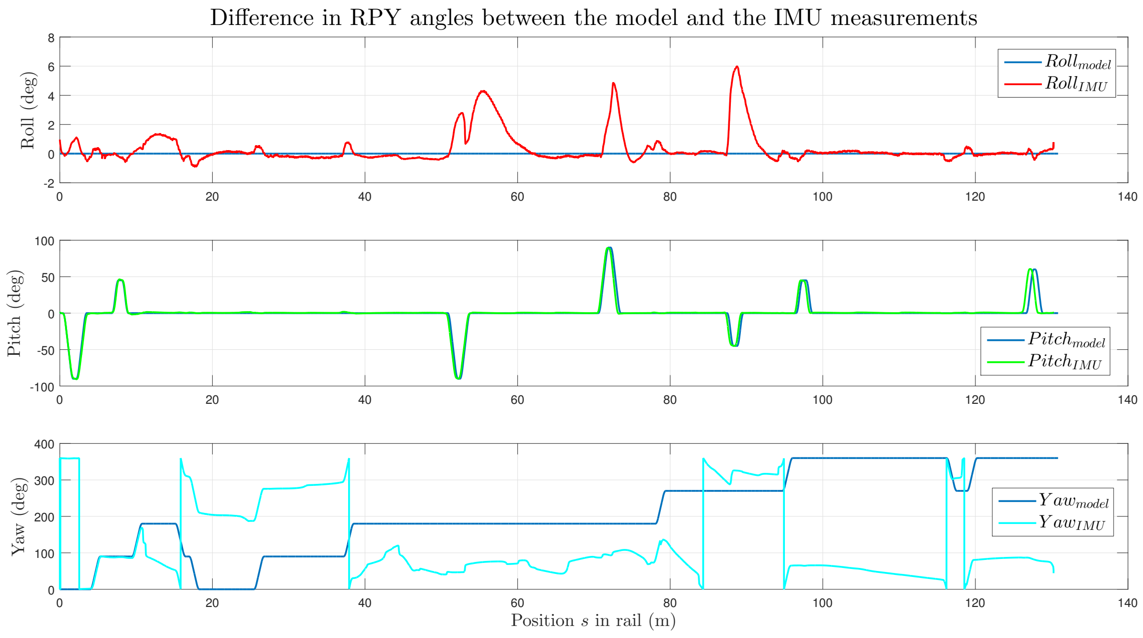


Figure 3.11: Differences between the modeled RPY values and the IMU measurements within a complete rail loop on CENPES rail.

It can be seen that there are roll variations between the model and the measurements mainly when the robot is on a vertical motion, but they are not greater than 6°. It is interesting to note that, as expected, even when the robot rolls on vertical tracks of the rail, the roll value decreases exponentially to zero due to the robot weight. The pitch measurement is actually fairly true to the model. The phase differences observed in the pitch graph of Figure 3.11 are due to the odometry error, as the IMU measurements are referenced to the odometric position in these graphs.

On the other hand, after the robot has travelled about only 11m, the IMU yaw graph gets totally distorted from the truth due to magnetic interference. Note that the sudden transitions in the yaw measurements are due to the discontinuity between 0° and 360°, as the measurements are wrapped into the interval $[0°, 360°)$.

Spatial OEM IMU offers the option of using other sources to infer the heading direction. One possibility is to estimate it from the direction of the linear velocity and acceleration. However, they have to be significant for a good estimate, that is, more than 2m/s in the case of the velocity, which is higher than DORIS maximum speed. Also, the linear velocity measurement is subject to GNSS availability. Other option is to use an external source to derive heading, as north seeking gyroscopes and reference markers, but this is not the case for DORIS.

Spatial OEM IMU also receives GNSS signals from the American GPS and Russian GLONASS satellite navigation systems. It also supports the Chinese BEIDOU and the European GALILEO, but these systems are not fully operational yet. GNSS systems provide absolute position and linear velocities with a precision of 2.5m in position and 0.05m/s in velocity (Advanced Navigation 2013). However, this accuracy is unacceptable for DORIS, and the GNSS receiver must have a clear signal from at least four satellites to work, while the CENPES utility plant is not an outdoor environment, being full of obstructions to the GNSS signals. For these reasons, GNSS measurements are not used for DORIS localization.

The IMU ends up providing only the pitch angle as a useful measurement for DORIS localization. Therefore, a probability model $p(\phi_t^{\mathrm{IMU}} \mid s_t, m)$ for the pitch measurement has to be derived. Actually, the pitch and yaw angular velocities are also used to detect in which type of curve the robot is, and that is useful in a proposed extension to the particle filter algorithm, described in Section 4.2. However, they will not be used directly as measurements in the perception step.

An appropriate way to model this probability is to represent it by a Gaussian centered on the expected pitch value derived from the mapping function $\phi_{bt} = f_{map}(s_t, m)$, where $\phi_{bt}$ is the base pitch angle. The standard deviation $\sigma^{\phi}$ of the Gaussian models the inaccuracy of the sensor. As the pitch values are wrapped to the interval $[-\pi, \pi)$, it is better to evaluate the difference $\phi_t^{\mathrm{IMU}} - \phi_{bt}$ in a Gaussian with zero mean and $\sigma^{\phi}$ standard deviation. This Gaussian returns the probability of measuring a pitch value of $\phi_t^{\mathrm{IMU}}$ given a position $s_t$, as $\phi_b$ is a function of $s_t$:

$$p(\phi_t^{IMU} \mid s_t, m) \sim \mathcal{N}\left(\phi_t^{IMU} - \phi_b(s_t); 0, {\sigma^{\phi}}^2\right) . \tag{3.12}$$

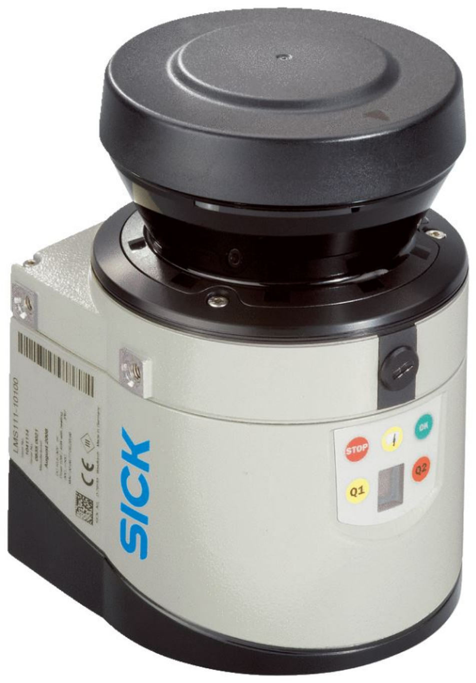
The pseudocode for the pitch measurement probability is given below.

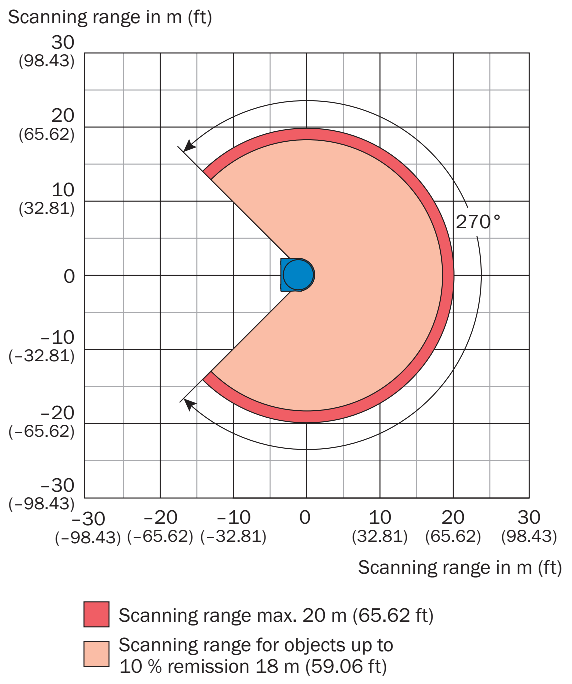
**Algorithm 10** IMU Pitch Probability

```
Require: $s_t$, $\phi_t^{IMU}$, $m$
Ensure: $p(\phi_t^{IMU} \mid s_t, m)$
 1: $\phi_{bt} = f_{map}(s_t, m)$
 2: $\Delta\phi_t = \phi_t^{IMU} - \phi_{bt}$
 3: Wrap $\Delta\phi_t$ to $[-\pi, \pi)$
 4: $p(\phi_t^{IMU} \mid s_t, m) \sim \mathcal{N}\left(\Delta\phi_t; 0, {\sigma^{\phi}}^2\right)$
 5: return $p(\phi_t^{IMU} \mid s_t, m)$
```

### 3.4.3 Laser Scanner Measurements

DORIS is also provided with a *SICK LMS111-10100* laser scanner. This model has an aperture angle of 270° and provides scan points with range and bearing data with a resolution of $0,25°$ at a rate of 25Hz. The maximum range of the scan is 20m and the sensor has an accuracy of a few centimeters (Figure 3.12).


(a)



(b)

Figure 3.12: (a) SICK LMS111-10100 laser scanner and (b) its scanning range.

The device is mounted on the rear part of the robot base in a way that the 90° blind region is always oriented to the external area of the rail loop. Thereby, the laser scans the process plant environment enclosed by the rail, which is important for DORIS mapping and monitoring tasks, but also for localization purposes. Three features of the rail and the environment were found to be interesting for localization and they are detectable in the laser scans: the floor, the rail geometry, and the rail fixation system. A probabilistic model for each of these three features is described in the following sections.

### Floor Scan Model

The rail installed in CENPES has four different height levels. Thus, a measurement of the distance from the laser scanner to the floor is a useful information to distinguish localization hypothesis about these heights. This distance is easily obtained by processing the laser scan, and can be compared to a model if one knows the robot position on the rail.

Considering the assumption A1 that the robot does not turn around the rail, and that the laser is mounted on the robot base with its $\vec{y}_{\text{laser}}$ axis pointing to the floor when the robot is on a horizontal section, it is easy to known which points in the laser scan correspond to the floor. Therefore, one way to estimate the distance $z^{floor}$ from the laser scanner to the floor in the $\vec{y}_{\text{laser}}$ direction is to specify a *Region of Interest* (RoI), select the points of the laser scan that are inside this region, and calculate the mean value of the ranges of these points in the $\vec{y}_{\text{laser}}$ component.

The RoI for the *floor points* was specified as a rectangle that goes 6m under the laser position and has a width of 0.2m. If the number of points situated inside this region is insufficient, which may be the case when the robot is on a vertical position, the algorithm truncates the range value to $z^{floor} = 6$m. Otherwise, the mean of the $\vec{y}_{\text{laser}}$ components of the floor points is calculated and returned as the variable $z^{floor}$. The truncation is made because ranges greater than the maximum rail height (4.7m) are inaccurate, as they are only given when the robot is approaching a vertical position, where the theoretical range tends to infinity.

Figure 3.13 shows a typical laser scan, the specified floor RoI, and the range value returned by the algorithm. For a more intuitive visualization, the scan points are disposed in the $y_b z_b$ robot base plane, as if the robot is viewed from the front.

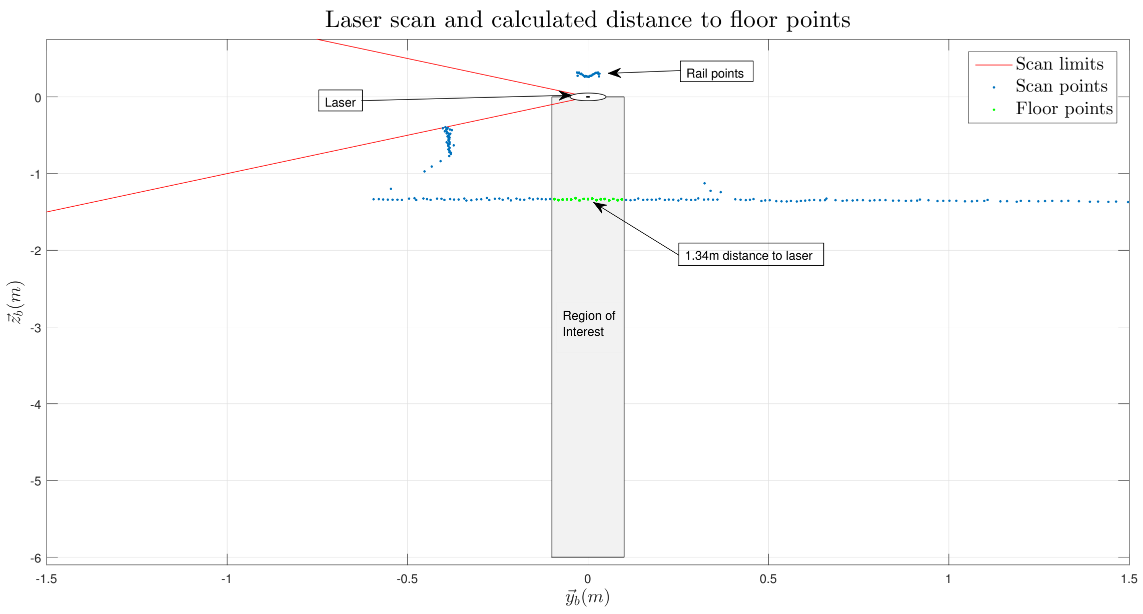


Figure 3.13: Typical laser scan with the identified floor points in green.

The laser distance to the floor can be modeled by simple geometric relations from the robot frontal gimbal position $s$, the base pitch angle $\phi_b$ (which is actually obtained from $s$), and some robot parameters, as shown in Figure 3.14.

The value $Z_1$ is obtained from the mapping function $Z_1 = f_{map}(s, m)$ and can also be expressed by the relation in (3.13), where $Z_m^{floor}$ is the modeled distance from the laser to the floor in the $\vec{y}_{\text{laser}}$ axis:

$$Z_1 = \frac{d}{2}\sin(\phi_b) + L_z\cos(\phi_b) + L_x\sin(\phi_b) + Z_m^{floor}\cos(\phi_b) \tag{3.13}$$

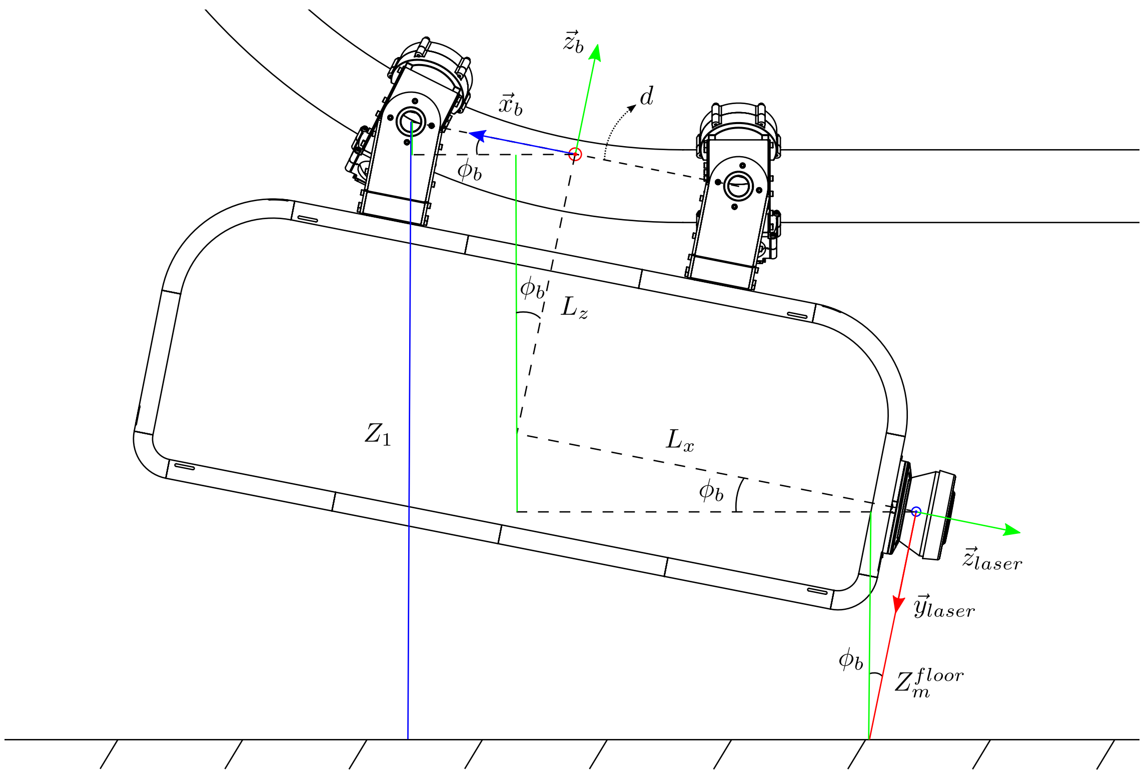


Figure 3.14: Geometric model of the floor scan.

This equation can be rearranged to express $Z_m^{floor}$ as a function of $Z_1$ and $\phi_b$:

$$Z_m^{floor} = \frac{Z_1}{\cos(\phi_b)} + \left(\frac{d}{2} + L_x\right)\tan(\phi_b) - L_z \tag{3.14}$$

$Z_m^{floor}$ goes to infinity when $\phi_b = \pm\pi/2$, as expected. Thus, $Z_m^{floor}$ is also truncated to 6m.

In specific parts of the rail path, there are obstructions between the robot and the floor, and the laser may detect points other than the floor points, corrupting the floor distance estimation. However, if one knows the positions and distances of these obstructions relative to the floor, they can be included in the model. In CENPES environment, there is a single situation where DORIS passes above electric panels and, thus, the algorithm returns the distance from the robot to the top of these panels. However, as the position and dimensions of these panels are known, they are included in the model.

The measurement model $p(z_t^{floor} \mid s_t, m)$ can then be computed as a Gaussian with mean equal to $Z_m^{floor}(s_t)$ and standard deviation $\sigma^{floor}$, which represents the uncertainty in the laser measurements and the model:

$$p(z_t^{floor} \mid s_t, m) \sim \mathcal{N}\left(z_t^{floor}; Z_m^{floor}(s_t), {\sigma^{floor}}^2\right) \tag{3.15}$$

The procedures to compute the floor measurement probability are listed in Algorithm  11.

**Algorithm 11** Floor Scan

```
Require: $s_t$, laser scan, $m$
Ensure: $p(z_t^{floor} \mid s_t, m)$
 1: Search for the laser scan points in the floor $RoI$
 2: Compute the mean $z_t^{floor}$ of the $\vec{y}_b$ components of these points
 3: if $s_t$ is in the region above the panels then
 4:     $Z_m^{floor} = Z_m^{\text{panel}}$
 5: else
 6:     $[Z_1, \phi_{bt}]^T = f_{map}(s_t, m)$
 7:     $Z_m^{floor} = \frac{Z_1}{\cos(\phi_{bt})} + \left(\frac{d}{2} + L_x\right)\tan(\phi_{bt}) - L_z$
 8: end if
 9: if $Z_m^{floor} > Z_{max}$ then
10:     $Z_m^{floor} = Z_{max}$
11: end if
12: $p(z_t^{floor} \mid s_t, m) \sim \mathcal{N}\left(z_t^{floor}; Z_m^{floor}, {\sigma^{floor}}^2\right)$
13: return $p(z_t^{floor} \mid s_t, m)$
```

Figure 3.15 compares the model with the laser readings of the distance to the floor. The observed phase difference is due to the odometry error, as the laser measurements are referenced to the odometric position in this graph.

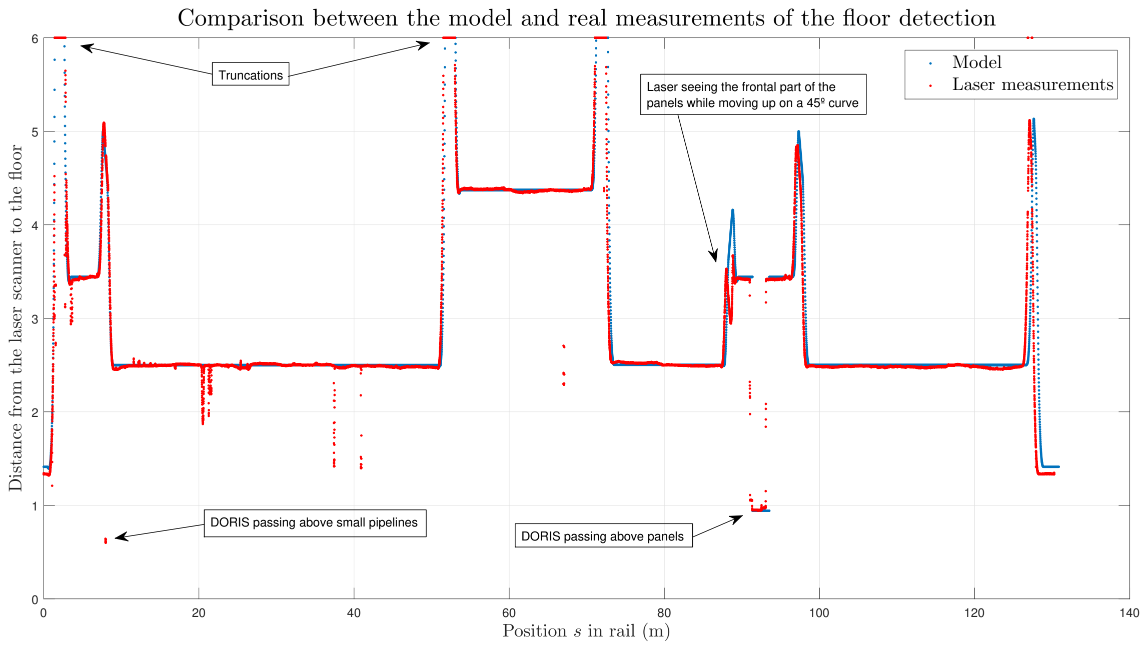


Figure 3.15: Comparison between the model and the laser readings of the floor.

One can see that the model is fairly true to the real laser measurements unless in regions of non-modeled obstructions, as when the robot passes above 50mm diameter pipelines or when DORIS is moving up on a 45° curve near electric panels. As these regions are very short relative to the entire rail, and considering that probabilistic algorithms deals well with non-modeled features, these differences can be neglected.

However, it is important to note that this measurement model has the disadvantage of being vulnerable to non-modeled obstructions. People walking under the robot, although they shouldn't due to safety reasons, and equipments under the rail path are some examples. The algorithm also may not be applicable for offshore environments, where some parts of the floor are made of gratings, and the floor may not be recognized in the laser scans. However, for onshore environments, as is the CENPES utility plant, the floor measurement is useful and is considered here.

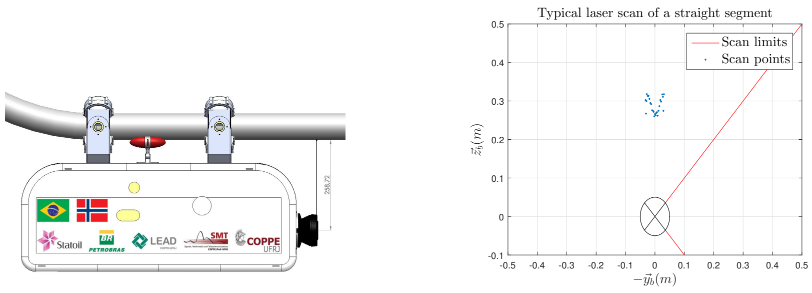
### Rail Scan Model

An interesting feature that can also be recognized by the laser scanner is the rail geometry. As the laser is mounted on the rear part of the robot, the positions of the scan points that correspond to the tubular rail geometry are different for curved and straight segments. As a consequence, each type of curve has a particular pattern that can be recognizable in the scans, as is shown in Figure 3.16. In this figure, the laser scans are plotted as if the robot was seen from the back for an intuitive comparison with the CAD figures on the left.

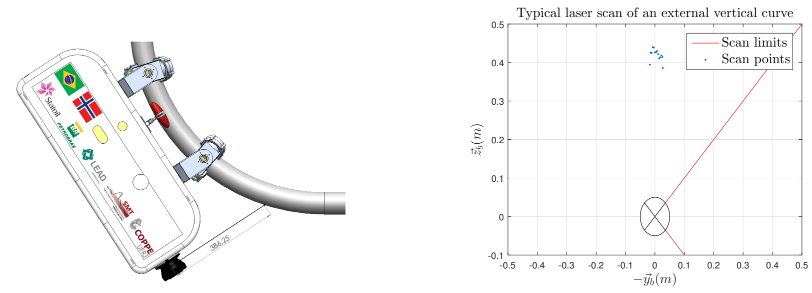
As can be noted in Figure 3.16, the information of the *rail points* in the scan is very useful for self-localization, as the robot can identify in which of the five types of rail segment it is. Therefore, a probabilistic model $p(z_t^{rail} \mid s_t, m)$ for the rail scan is also derived to be integrated with the other measurements in the localization algorithm.

The detection of the type of rail section the robot is on could, in principle, be given by a simpler algorithm that uses the angular velocities measured by the IMU and the linear velocity provided by the motion control. However, the laser approach has the advantage of recognizing the section type through the rail geometry observed by the robot, instead of the robot motion. This implies that, even when DORIS is stationary on a curved part of the rail, the laser scan algorithm can reliably detect which curve type it is, while the same does not occur for the other approach.

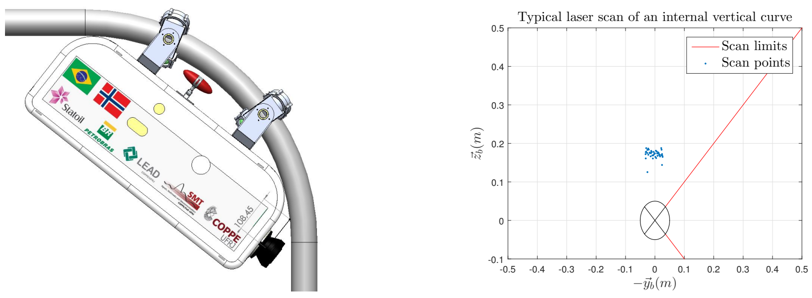
The position of the rail points relative to the laser frame can be modeled with simple trigonometry, as in the floor scan model. To simplify the *rail scan model*, one can summarize the positions of all the rail points detected by the laser by the mean $z^{rail}$ of these points given in the robot base $y_b z_b$ plane. In this way, one can work with a single point, instead of a set of points, with reasonable accuracy. If one knows the robot $s_t$ position and the position $s_r$ of this single rail point, the transformation between the laser frame and the rail frame corresponding to this point can be obtained. Then, the expected position $\mu^{rail}$ of the rail point in the laser scan ($y_b z_b$ plane) can be calculated.



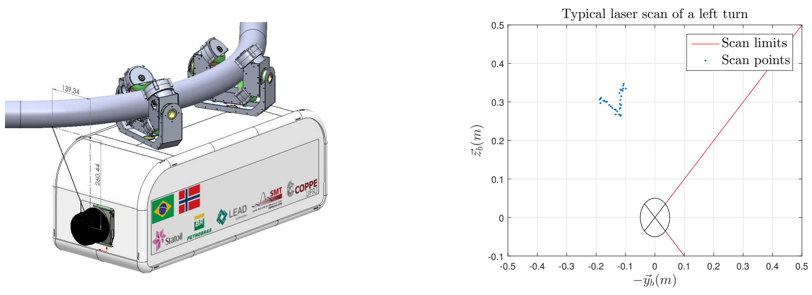
(a) Straight section.



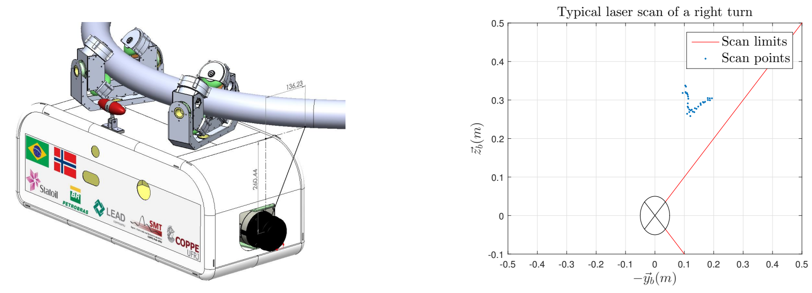
(b) External vertical curve.



(c) Internal vertical curve.



(d) Left turn.



(e) Right turn.

Figure 3.16: Typical laser scans for the five different types of rail segments.

Considering that the robot does not move in two curves at the same time (assumption A2), each rail section type can be analyzed separately. For the straight section, it is trivial to retrieve the position of the rail point in the scan: it is directly above the laser frame with the distance of $L_z - \varnothing_{rail}/4$ in the $\vec{z}_b$ direction, where $\varnothing_{rail}$ is the rail tube diameter.

For the curved sections, we have to estimate the parametric position of the rail point identified by the laser, as it cannot be directly inferred from the geometric model. Considering Figure 3.17, the arch length $\Delta s$ between the rear gimbal position $s_2$ and the rail point position $s_r$ can be approximated according to (3.16). This is the best approximation one can get from the geometric model.

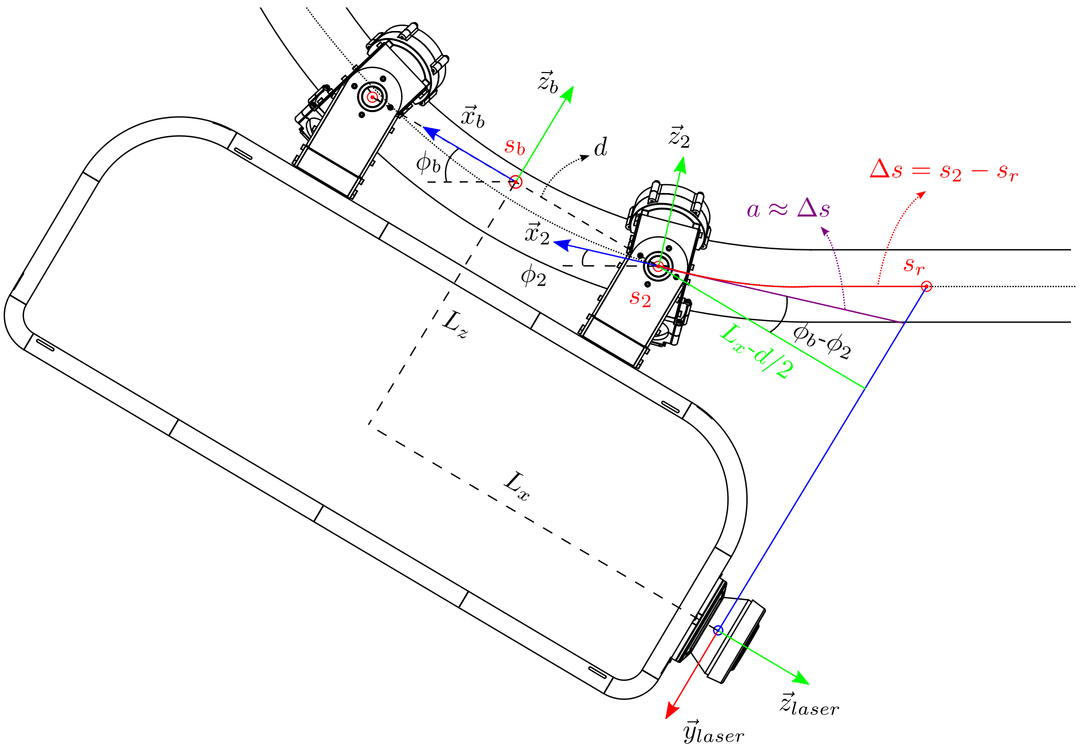


Figure 3.17: Geometric model of the rail detection for the vertical case.

$$a = \frac{L_x - \frac{d}{2}}{\cos(\alpha_b - \alpha_2)}$$
$$a \approx \Delta s = s_2 - s_r$$
$$\Delta s \approx \frac{(L_x - \frac{d}{2})}{\cos(\alpha_b - \alpha_2)}, \tag{3.16}$$

where $\alpha$ is $\phi$ for vertical curves and $\theta$ for horizontal curves. The figure of the horizontal case was omitted here, but the same relation is obtained.

In this formulation, it is supposed that the robot roll angle is zero (assumption A1). Noting that at least one of the two statements $\phi_b = \phi_2$ and $\theta_b = \theta_2$ is always true, due to assumption A2, equation (3.16) can be rewritten as:

$$\Delta s \approx \frac{(L_x - \frac{d}{2})}{\cos(\phi_b - \phi_2)\cos(\theta_b - \theta_2)} \,. \tag{3.17}$$

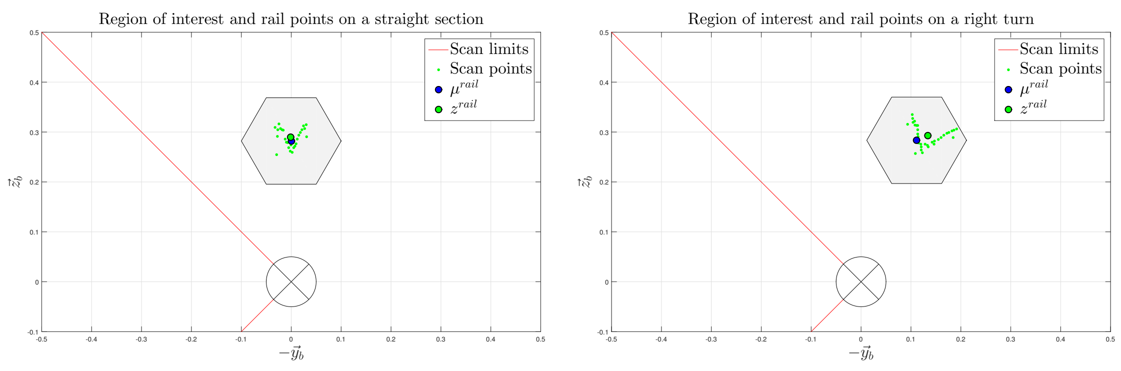
One should note that the denominator will never be zero, as there is no situation in the DORIS rail where $\phi_b - \phi_2$ or $\theta_b - \theta_2$ equals to $\pm\pi$.

After calculating the rail point parametric position by $s_r = s_2 - \Delta s$, its Cartesian position is mapped through $[X_R, Y_R, Z_R]^T = f_{map}(s_r, m)$, and the translation vector between the laser frame and this position is obtained. If $\phi_b$ and $\theta_b$ are known, this vector can be represented in the laser scan plane.

Identifying the points in the scan that correspond to the rail geometry is not as easy as detecting the floor points, for example. However, it can be done by searching points near the expected rail point $\mu^{rail}$, given by the model described above. Therefore, a RoI centered on the expected rail point $\mu^{rail}$ is defined, and the scan points inside this RoI are considered as the rail points. For a computationally simple search of these points, the RoI was defined as an hexagon centered at $\mu^{rail}$ with side $L_{\mathrm{RoI}}$, defined after field tests analysis. The average position of the points inside the RoI is then considered to be the measurement $z^{rail}$.

If there are insufficient number of points inside the RoI, the algorithm considers that $z^{rail}$ is on the RoI boundary. In this case, a gain $\gamma$, with $0 \leq \gamma \leq 1$, is considered to decrease the probability in (3.18). This gain represents that the algorithm has failed to find rail points, and, so, the probability should be decreased.

In Figure 3.18, the RoIs of the laser scans acquired when the robot was in a straight section (Figure 3.18(a)) and right turn (Figure 3.18(b)) are illustrated, highlighting the rail point $\mu^{rail}$ given by the model, and the measurement $z^{rail}$.



(a) Straight section (b) Right turn

Figure 3.18: Rail points inside RoIs for (a) a straight section and (b) a right turn.

Figure 3.19 shows the comparison of the model with field test measurements corresponding to a complete loop in the CENPES rail. It can be seen that the model is fairly true to the real measurements. The observed amplitude differences are due to the approximations considered in the model, and there are phase differences between the graphs because these measurements are associated to the odometry.

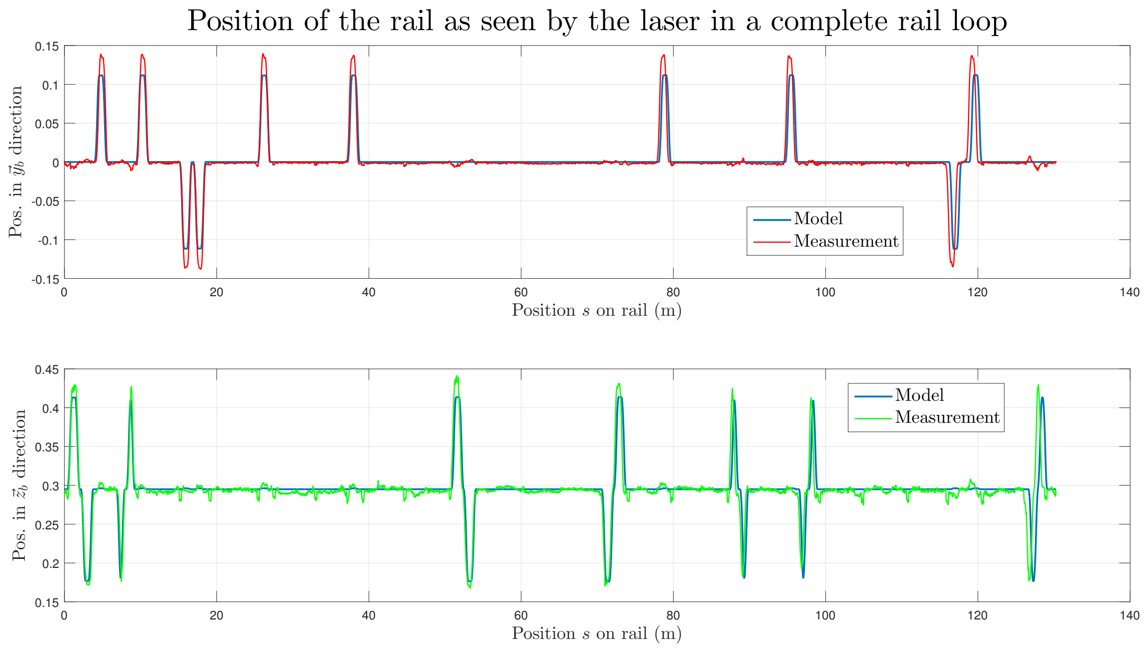


Figure 3.19: Comparison between the model and the real measurements of the rail position as seen by the laser scanner.

The *rail scan probability* $p(z_t^{rail} \mid s_t, m)$ is calculated through the evaluation of $z_t^{rail}$ in a multivariate Gaussian centered at $\mu_t^{rail}$ with covariance $\Sigma^{rail}$, which models the incorrectness of the model and the measurement noise. This measurement probability is expressed as:

$$p(z_t^{rail} \mid s_t, m) \sim \mathcal{N}(z_t^{rail}; \mu_t^{rail}(s_t, m), \Sigma^{rail}) \tag{3.18}$$

This Gaussian is two dimensional, as the points are evaluated in the scan plane, and, thus, the mean $\mu_t^{rail}$ is a vector with two components. The covariance matrix ${\Sigma^{rail}}^2$ is a two dimensional diagonal matrix with $\sigma^{rail}$ as the diagonal elements, as it is supposed that the uncertainty is the same in both dimensions. The rail scan algorithm is given in Algorithm  12.

### Pole Detection Model

Another feature that is recognizable on the laser scans is the rail fixation system. This system is composed of tubular poles and angle bracket structures that support the rail from the top, suspending it. There are two types of fixations on the CENPES rail: angle brackets attached to poles fixed on the floor (Figs. 3.2 and 3.7), and just brackets fixed to beams. After analysis of the laser scan pattern throughout the rail loop, it was found that some of these fixations are very distinguishable in the scan, and, as their positions are known, they are useful for localization.

**Algorithm 12** Rail Scan

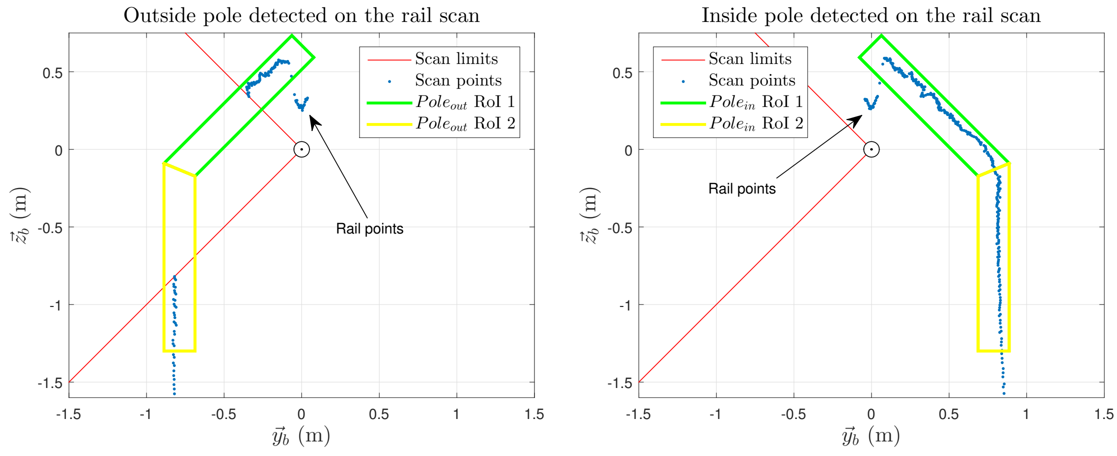
```
Require: $s_t$, laser scan, $m$
Ensure: $p(z_t^{rail} \mid s_t, m)$
 1: $[\phi_2, \theta_2]^T = f_{map}(s_t, m)$
 2: $[X_b, Y_b, Z_b, 0, \phi_b, \theta_b]^T = f_{map}(s_t, m)$
 3: $[X_{laser}, Y_{laser}, Z_{laser}]^T = R_{b,laser}^T [X_b, Y_b, Z_b]^T$
 4: $\Delta s = \left(L_x - \frac{d}{2}\right) / [\cos(\phi_{bt} - \phi_{2t}) \cos(\theta_{bt} - \theta_{2t})]$
 5: $[X_R, Y_R, Z_R]^T = f_{map}(s_t - d - \Delta s, m)$
 6: $P_{laser,R} = [X_R, Y_R, Z_R]^T - [X_{laser}, Y_{laser}, Z_{laser}]^T$
 7: Represent $P_{laser,R}$ in the robot base $yz$ plane and compute $\mu^{rail}$ by decreasing
    the magnitude of $(P_{laser,R})_{base}$ by $\varnothing_{rail}/4$
 8: Search for scan points in the rail RoI centered at $\mu^{rail}$ and get the mean $z_t^{rail}$ of
    these points
 9: if there are insufficient points in the RoI then
10:     Consider $z_t^{rail}$ at the RoI boundary
11:     $p(z_t^{rail} \mid s_t, m) \sim \gamma \mathcal{N}(z_t^{rail}; \mu_t^{rail}(s_t, m), \Sigma^{rail})$
12: else
13:     $p(z_t^{rail} \mid s_t, m) \sim \mathcal{N}(z_t^{rail}; \mu_t^{rail}(s_t, m), \Sigma^{rail})$
14: end if
15: return $p(z_t^{rail} \mid s_t, m)$
```

The detectable parts of the fixation system are then included in the map (Figure 3.4), and a probabilistic model for their detection is formulated. It is pretty simple and somewhat similar to the red landmark detection model. We are interested here in deriving the probability distribution $p(z_t^{pole} \mid s_t, m)$, and, to do so, we have to associate a measurement $z_t^{pole}$ with information of the model provided by the robot position $s_t$ and the map $m$.

Within the recognizable fixation system parts, two types of features can be distinguishable, as there are poles positioned inside the rail loop region and others on the external part. Therefore, they are on different sides in the rail scan and can be seen as distinct features. A model for the *pole type detection* is created by associating each type of pole with its respective position $s_i^{pole}$ on the rail. Type *one* is defined as the poles inside the loop, while type *two* are the external ones, and type *zero* is the event of not detecting any pole in the scan.

To detect a pole type in the laser scan, a procedure similar to the floor scan model is used. Regions of interest to search for*pole points* are defined according to the poles geometry and positions relative to the laser scanner, which are given in the CAD model. There are two RoIs for each pole type, one corresponding to the pole tubular geometry and the other to the angle bracket geometry. If there are enough scan points in both the two RoIs, then a pole type is detected. The minimum number of scan points was defined after several analysis of the rail scans acquired in field tests. With the defined parameters, the robot is able to detect all the considered poles of the map even when moving at its highest speed.

The external pole RoIs are obtained by simply mirroring the internal pole RoIs about the robot right plane. Figure 3.20(b) shows the detection of a pole that is inside the rail loop and Figure 3.20(a) a pole in the external area.



(a) Detection of a pole outside the rail loop. (b) Detection of a pole inside the rail loop.

Figure 3.20: Detection of poles on the scan: (a) outside, and (b) inside poles.

With the detected pole type, the robot position $s_t$, and the positions of all the recognizable poles, the probability $p(z_t^{pole} \mid s_t, m)$ is calculated through:

$$p(z_t^{pole} \mid s_t, m) \sim \mathcal{N}\left(s_t - s_i^{pole}; 0, {\sigma^{pole}}^2\right) , \tag{3.19}$$

where $s_i^{pole}$ is the closest pole (of the detected type) position to $s_t$ and $\sigma^{pole}$ is the standard deviation used to accommodate small errors in the model, as the poles positions and the total lap length uncertainties. Naturally, the probability of not detecting any pole at a position $s_t$ is obtained by $p(z_t^{no\_pole} \mid s_t, m) = 1 - p(z_t^{in\_pole} \mid s_t, m) - p(z_t^{out\_pole} \mid s_t, m)$. In this model, it is assumed that two poles are not detected at the same time, which is true, as no more than one fixation is used at the same place of the rail. The *pole detection algorithm* is found in Algorithm 13.

The probability densities of inside and outside pole detections according to the model over all the rail positions within a loop are shown in Figure 3.21.

## 3.4.4 Other Measurement Possibilities

Besides of using a camera to detect artificial landmarks, IMU gyroscopes to provide pitch angle data, and a laser scanner to detect natural landmarks, other measurement possibilities were investigated.

DORIS has an RGB camera pointed to the inspected environment that could be used in a visual odometry algorithm, where the direction of motion and the displacement between two frames would be estimated. The camera images could

**Algorithm 13** Pole Detection

```
Require: $s_t$, laser scan, $m$
Ensure: $p(z_t^{pole} \mid s_t, m)$
 1: Search for scan points in the in_RoIs and out_RoIs
 2: if num. of points inside the in_RoIs > threshold then
 3:     $pole = in\_pole$
 4: else if num. of points inside the out_RoIs > threshold then
 5:     $pole = out\_pole$
 6: else
 7:     $pole = no\_pole$
 8: end if
 9: Search for the nearest poles $s_i^{in\_pole}$ and $s_j^{out\_pole}$ on the rail map $m$
10: $p(z_t^{in\_pole} \mid s_t, m) \sim \mathcal{N}\left(s_t - s_i^{in\_pole}; 0, {\sigma^{pole}}^2\right)$
11: $p(z_t^{out\_pole} \mid s_t, m) \sim \mathcal{N}\left(s_t - s_i^{out\_pole}; 0, {\sigma^{pole}}^2\right)$
12: if $pole = in\_pole$ then
13:     $p(z_t^{pole} \mid s_t, m) = p(z_t^{in\_pole} \mid s_t, m)$
14: else if $pole = out\_pole$ then
15:     $p(z_t^{pole} \mid s_t, m) = p(z_t^{out\_pole} \mid s_t, m)$
16: else
17:     $p(z_t^{pole} \mid s_t, m) = 1 - p(z_t^{in\_pole} \mid s_t, m) - p(z_t^{out\_pole} \mid s_t, m)$
18: end if
19: return $p(z_t^{pole} \mid s_t, m)$
```

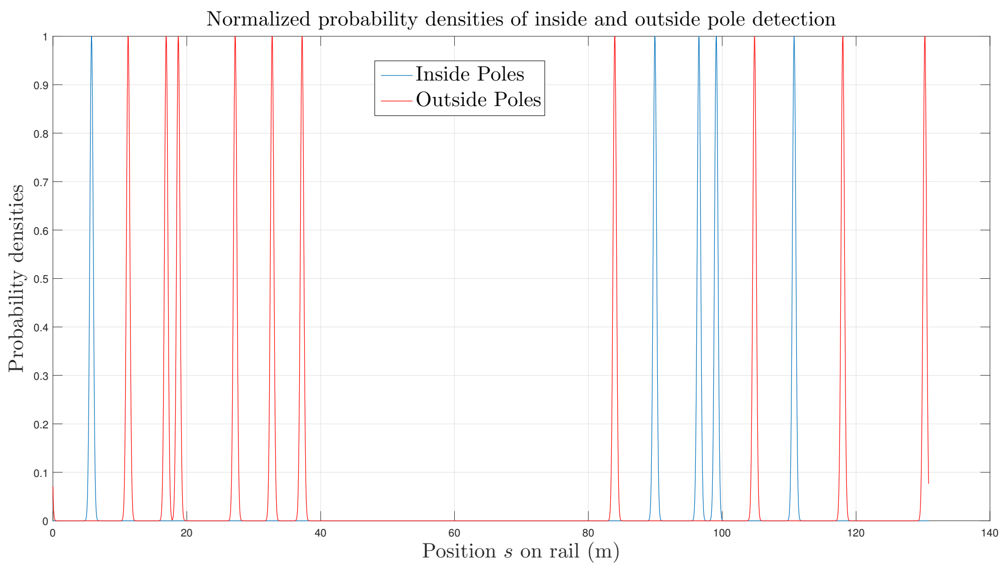


Figure 3.21: Probability densities of pole detection over the position on the rail.

also be matched to a reference sequence of a complete loop on the rail, where each frame would be associated to a rail position. However, these algorithms would be computationally expensive and probably would work only when the robot is travelling relatively slow.

Other types of artificial landmarks could be placed on the rail and detected by the robot, such as QR codes (as mentioned in 3.4.1), RFID tags, and magnets with hall effect sensors. However, the option for visual landmarks was the simplest one in terms of implementation and computational effort, besides of being detectable even when the robot is moving fast.

DORIS IMU also provides acceleration, magnetic, and GPS data, besides of orientation values. As in García (2012), the acceleration component on the along-track direction $\vec{x}$ could estimate motion, while the other two components could be useful to estimate rail curvatures. However, as DORIS motion is relatively slow compared to other implementations, such as cars and trains, the *signal to noise ratio* on the acceleration measurements would be considerable.

Magnetic data acquired by the IMU was analyzed and a consistent magnetic map of the environment during a rail loop was found when DORIS was slowly moving. This information could be useful in localization, as in Christensen, Fischer, Kroffke, Lemburg & Ahlers (2011). However, the robot motors unfortunately caused considerable magnetic interference when DORIS was travelling faster, and eventual operation of some equipments of the utility plant also caused this effect, disturbing the magnetic map. While the motors magnetic interference could possibly be compensated, the disturbances caused by the local equipments are unpredictable and uncontrollable.

Finally, GNSS data was also acquired in field tests and analyzed for localization. Unfortunately, the robot was not able to receive data from enough satellites to provide absolute position information, possibly due to the cluttered aspect of the utility plant environment. Moreover, the GNSS uncertainty of 2m probably would be too much to give any improvement to the localization algorithm.

### 3.4.5 Probabilistic Perception Model

DORIS *measurement probability* $p(z_t \mid x_t)$ is the probability of observing the measurements $z_t = \{z_t^{red}, \phi_t^{IMU}, z_t^{floor}, z_t^{rail}, z_t^{pole}\}$ given a rail position $s_t$ and the map $m$, as expressed below:

$$p(z_t \mid s_t, m) = p(z_t^{red}, \phi_t^{IMU}, z_t^{floor}, z_t^{rail}, z_t^{pole} \mid s_t, m) \tag{3.20}$$

Assuming that each measurement is independent from another, they are conditionally independent on $s_t$ and $m$. Therefore, (3.20) can be expressed as:

$$\begin{aligned} p(z_t \mid s_t, m) \ &= \ p(z_t^{red} \mid s_t, m) \ . \ p(\phi_t^{IMU} \mid s_t, m) \ . \ p(z_t^{floor} \mid s_t, m) \ . \\ & \quad\ p(z_t^{rail} \mid s_t, m) \ . \ p(z_t^{pole} \mid s_t, m) \end{aligned} \tag{3.21}$$

This probability is used in DORIS particle filter algorithm, which is described in Section 4.1, to compute the particle importance weights. The individual probabilities are calculated through equations (3.11), (3.12), (3.15), (3.18), and (3.19).

# Chapter 4

# DORIS Localization

Considering the algorithms presented in Chapter 2, the particle filter (or Monte Carlo Localization, when it is implemented for localization) is the most suitable method for the DORIS localization problem. This is due to the following reasons:

- **Nonlinearities of the motion model**: DORIS moves suspended on a rail by two gimbal mechanisms fixed to a base. So, its motion is conditioned to a map of the rail, as the gimbals poses are equivalent to the rail track points poses. Although the gimbal motion is linear on the rail parametrization space, the robot base pose is not, specially when represented in the Cartesian space.
- **Nonlinearities of the measurement model**: DORIS measurements are conditioned on the characteristics of the rail and the surrounding environment, but most of them, as the laser scans, are given in the Cartesian space. The transformation of the robot position in the rail parametrization space to the Cartesian space is nonlinear. As mentioned in Section 2.5, nonparametric filters can deal with nonlinearities in the model.
- **Multimodal representation of beliefs**: several sections of DORIS rail have the same characteristics, such that a set of measurements may not distinguish multiple hypothesis of positions on the rail. Furthermore, DORIS uses landmark detection as measurements, but the correspondences are unknown. Therefore, a multimodal representation, that also deals with unknown correspondences, as MCL, is more appropriate.
- **Global localization and recovery from failures**: in order to increase the robot autonomy, it must be capable of finding itself in situations of complete confusion about its current position on the rail. Also, the robot must recover from failures, which could reset the robot position or keep it constant while the robot is moving, characterizing a kidnapped robot problem. Particle filters are capable of solving both problems.

In the following sections, the particle filter implementation for DORIS localization and a modification on the PF proposal distribution, similar to the one found in Lenser & Veloso (2000), are presented. Basically, this modification resets particles to positions that are consistent to the recent events observed by the robot. This extension of the algorithm was proved to substantially increase the filter performance, solving the global localization and kidnapped robot problems, and providing fast convergence of the estimation with only a small number of particles.

## 4.1 DORIS Particle Filter

The implementation of MCL for DORIS is pretty straightforward, given that the particle filter steps were already presented in Section 2.5, and the motion and perception models were established in Chapter 3. Considering that the map is known, the DORIS localization problem is that of estimating the posterior belief of the single state $s_t$ derived in (4.1) on a similar manner to (2.19), with the addition of the map information:

$$\begin{aligned}
bel(s_{0:t}) &= p(s_{0:t} \mid z_{1:t}, u_{1:t}, m) \\
&\overset{\text{Bayes}}{=} \eta p(z_t \mid s_{0:t}, z_{1:t-1}, u_{1:t}, m) p(s_{0:t} \mid z_{1:t-1}, u_{1:t}, m) \\
&\overset{\text{Markov}}{=} \eta p(z_t \mid s_t, m) p(s_{0:t} \mid z_{1:t-1}, u_{1:t}, m) \\
&= \eta p(z_t \mid s_t, m) p(s_t \mid s_{0:t-1}, z_{1:t-1}, u_{1:t}, m) p(s_{0:t-1} \mid z_{1:t-1}, u_{1:t}, m) \\
&\overset{\text{Markov}}{=} \eta p(z_t \mid s_t, m) p(s_t \mid s_{t-1}, u_t, m) p(s_{0:t-1} \mid z_{1:t-1}, u_{1:t-1}, m) \\
&= \eta \overbrace{p(z_t^{red} \mid s_t, m) p(\phi_t^{IMU} \mid s_t, m) p(z_t^{floor} \mid s_t, m) p(z_t^{rail} \mid s_t, m) p(z_t^{pole} \mid s_t, m)}^{\textbf{measurement probabilities}} \\
&\quad \underbrace{p(s_t \mid s_{t-1}, u_t, m)}_{\textbf{state transition probability}} \underbrace{p(s_{0:t-1} \mid z_{1:t-1}, u_{1:t-1}, m)}_{bel(s_{0:t-1}) \;\to\; \textbf{recursion}}
\end{aligned} \tag{4.1}$$

Note that, in this derivation, the state is given as a sequence over time to simplify notation, but this does not lose generality. However, the particles were actually implemented to represent only the most current state estimation, and not all the particle history, as this would be computationally unfeasible.

As explained in Section 2.5.1, in a particle filter, the posterior belief is represented by a set of particles $\mathcal{X}_{0:t} = \{x_{0:t}^1, x_{0:t}^2, \ldots, x_{0:t}^N\}$. As the particles cannot be sampled directly from the posterior, they are drawn from a proposal distribution $g(x_{0:t}^n)$ and weighted proportionally to the fraction of the posterior function $p(x_{0:t}^n)$ over the proposal, which is represented as follows:

$$x_{0:t}^n \sim g(s_{0:t} \mid z_{1:t}, u_{1:t}, m) \tag{4.2}$$

$$w_t^n = \frac{p(s_{0:t} \mid z_{1:t}, u_{1:t}, m)}{g(s_{0:t} \mid z_{1:t}, u_{1:t}, m)} \tag{4.3}$$

The proposal distribution should be calculated in a recursive way. Therefore, it is factorized as:

$$g(s_{0:t} \mid z_{1:t}, u_{1:t}, m) = g(s_t \mid s_{0:t-1}, z_{1:t}, u_{1:t}, m) \underbrace{g(s_{0:t-1} \mid z_{1:t-1}, u_{1:t-1}, m)}_{\textbf{recursion}} . \tag{4.4}$$

The proposal function can be defined in different ways, but it should be appropriate to approach the true posterior and spread particles near the true state. As is usual in particle filter applications, the proposal $g(s_t \mid s_{0:t-1}, z_{1:t}, u_{1:t}, m)$ is chosen to be the motion model of (4.5) in order to spread particles according to the state transition density given by (3.10) regarding on a previous state. The samples can be easily drawn from the proposal density using (3.9).

$$g(s_t \mid s_{0:t-1}, z_{1:t}, u_{1:t}, m) \quad := \quad \overbrace{p(s_t \mid s_{t-1}, u_t, m)}^{\textbf{state transition probability (3.10)}} \tag{4.5}$$

$$g(s_{0:t} \mid z_{1:t}, u_{1:t}, m) = p(s_t \mid s_{t-1}, u_t, m) \underbrace{g(s_{0:t-1} \mid z_{1:t-1}, u_{1:t-1}, m)}_{\text{recursion}} \tag{4.6}$$

The particles importance factors incorporate the measurement data in the state estimation. They are calculated by the relation in (4.3) using the proposal distribution of (4.6) and the posterior from (4.1):

$$\begin{aligned} w_t^n &= \frac{\eta p(z_t \mid s_t, m) p(s_t \mid s_{t-1}, u_t, m)}{p(s_t \mid s_{t-1}, u_t, m)} \overbrace{\left( \frac{p(s_{0:t-1} \mid z_{1:t-1}, u_{1:t-1}, m)}{g(s_{0:t-1} \mid z_{1:t-1}, u_{1:t-1}, m)} \right)}^{\text{recursion } = \, w_{t-1}^n} \\ &\propto p(z_t^{red} \mid s_t, m) p(\phi_t^{IMU} \mid s_t, m) p(z_t^{floor} \mid s_t, m) p(z_t^{rail} \mid s_t, m) p(z_t^{pole} \mid s_t, m) w_{t-1}^n \end{aligned} \tag{4.7}$$

The individual particle weights are calculated disregarding the normalization constant $\eta$, at first. Subsequently, they are normalized by $\eta = (\sum_{n=1}^{N} w_t^n)^{-1}$. Note that the new importance factors $w_t^n$ keep the history of the particle weights (until a resampling step is done) by including $w_{t-1}^n$ in their calculations.

After sampling particles and computing weights to them, it is verified if resampling must be done by calculating the *effective sample size* with equation (2.22). When this value is below a fixed empirical threshold $N_{thres}$, a resampling procedure is performed through the *resampling wheel algorithm*, described in Algorithm 4. If resampling is performed, all particle weights are reset to $1/N$. Otherwise, the weights calculated in recursion are kept, according to equation (2.21).

To start the recursive PF algorithm, an estimation of the initial state $s_0$ has to be given. This requires the initialization of particles $w_0^n$ sampled from $bel(s_0)$. If one knows the exact position $\mu_0$ of the robot on the rail, all $N$ particles must be initialized as $\mu_0$. If the robot position is partially known around $\mu_0$ with some uncertainty $\sigma_0$, then the particles are sampled from the Gaussian $\mathcal{N}(s_0; \mu_0, {\sigma_0}^2)$. However, if the initial condition is completely unknown, then the particles are randomly distributed over all the rail track. DORIS particle filter is described in Algorithm 14.

**Algorithm 14** DORIS Particle Filter

```
Require: $\mathcal{X}_{t-1}$, $\mathcal{W}_{t-1}$, $z_t$, $u_t = \{v_t, \Delta T\}$, $m$
Ensure: $\mathcal{X}_t$, $\mathcal{W}_t$
 1: if $v_t$ is too small then
 2:     $\mathcal{X}_t = \mathcal{X}_{t-1}$
 3:     $\mathcal{W}_t = \mathcal{W}_{t-1}$
 4:     return $\mathcal{X}_t$, $\mathcal{W}_t$
 5: end if
 6: $\mathcal{X}_t = \mathcal{W}_t = \emptyset$
 7: for $n = 1$ to $N$ do
 8:     sample $x_t^n \sim p(x_t \mid x_{t-1}^n, u_t)$ (3.9)
 9:     $w_t^n = p(z_t \mid x_t^n) w_t^{n-1}$ (4.7)
10:     Append $x_t^n$ to $\mathcal{X}_t$
11:     Append $w_t^n$ to $\mathcal{W}_t$
12: end for
13: Normalize $\mathcal{W}_t$
14: Calculate the effective sample size $N_{eff}$ by (2.22)
15: if $N_{eff} < N_{thres}$ then
16:     Resample (Algorithm 4)
17:     $\mathcal{W}_t = \{1/N, 1/N, \ldots, 1/N\}$
18: end if
19: return $\mathcal{X}_t$, $\mathcal{W}_t$
```

### 4.1.1 Simulation Results

In this section, simulation results of the particle filter algorithm implemented for DORIS localization using experimental data acquired in field tests are presented. The tests were done with known initial and final positions, usually being started on the rail zero point and ended on the same position after one complete loop. Different velocities were considered to test the algorithm robustness to different motion conditions. All the results are compared to the odometric localization system, which uses the assumption that the robot initial state is known.

The rail map was built following the procedure described in Section 3.1 using a sample resolution of 1cm over the 130.85m of rail length installed in CENPES. The parameters in Table 4.1 were established for all the simulations presented in this work. These parameters were defined after several analysis of the data acquired in the field tests.

| $\alpha_{odom}$ | $red_{\text{thres}}$ | $\sigma^{red}$ | $\sigma^{\phi}$ | $\sigma^{floor}$ | $\sigma^{rail}$ | $\sigma^{pole}$ | $N_{thres}$ |
|---|---|---|---|---|---|---|---|
| 5% | 75% | 0.3m | 5° | 0.25m | 0.1m | 0.25m | 75%N |

Table 4.1: Table of parameters considered for the simulations.

**Simulation 1: tracking**

In this first simulation, the localization algorithm is tested for a tracking problem, in which the robot initial position is $s_0 = 0$m, which corresponds to $[X, Y, Z]^T = [0, 0, 0]^T$, and it completes a rail loop moving on the positive direction (CCW) with constant velocity $v = 0.6$m/s. The robot's belief is initialized as a Gaussian centered at $s_0 = 0$ with standard deviation $\sigma_0 = 1$m, and 50 particles are drawn from this distribution, as shown in Figure 4.1(a).

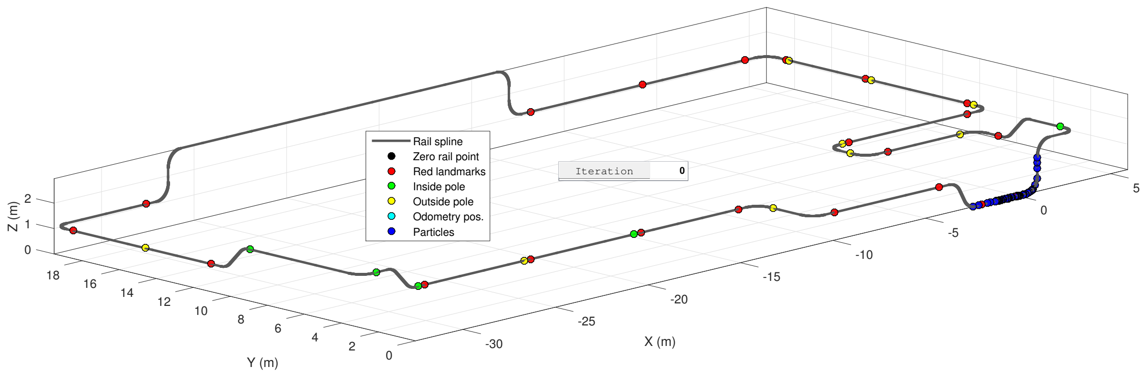
After a single iteration of the algorithm, the particles that are not at the same height of the robot, and the ones that are in curves, receive very low weights, so they are replaced in the resampling step. In a further iteration, the robot senses a red landmark and now the only surviving particles are on the regions near the two red markers of the rail base (Figure 4.1(b)). The robot moves further, carrying on two hypothesis about its belief, when it finally reaches a curved section. The height and rail geometry observed by the robot when it is on the curve makes one of the hypothesis (that is still on the base) unlikely, and the corresponding particles are eliminated after resampling is performed (Figure 4.1(c)). At this stage, the robot has successfully localized itself and it carries particles near the true state until the end of the test (Figure 4.1(d)).

The average value of the particles positions compared to the rail length (which is supposed to be the actual travelled distance) reveals an error of -18.25cm, while the odometry estimate has an error of -59cm, due to wheel slipping and sliding.

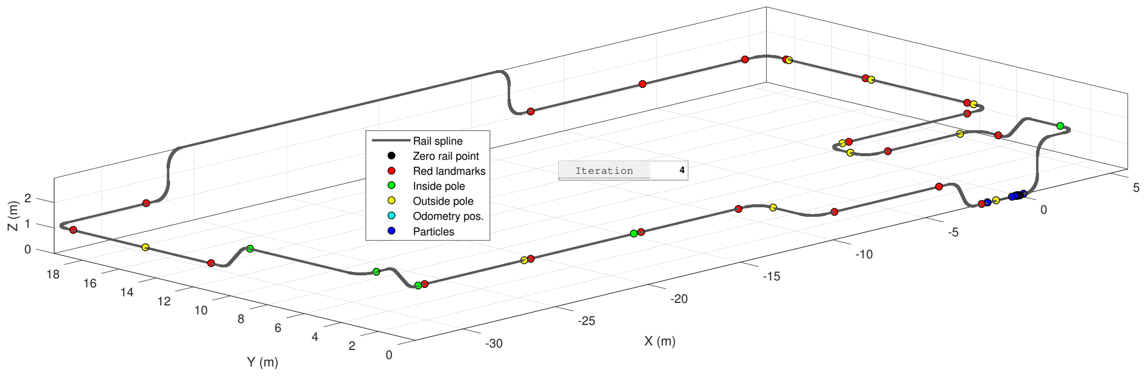
**Simulation 2: tracking with different motion patterns**

While the odometry error observed after a complete loop in Simulation 1 is not so significant, Simulation 2 analyzes a situation where this error is very large. After several field tests, it was evidenced that the wheels slip when the robot is moving upwards and slide when it is descending. The slipping issue is due to the difficulty that the motor controllers have of moving the robot on the opposite direction of gravity, while the sliding is due to the controllers trying to brake the robot when it is moving on the same direction of gravity.

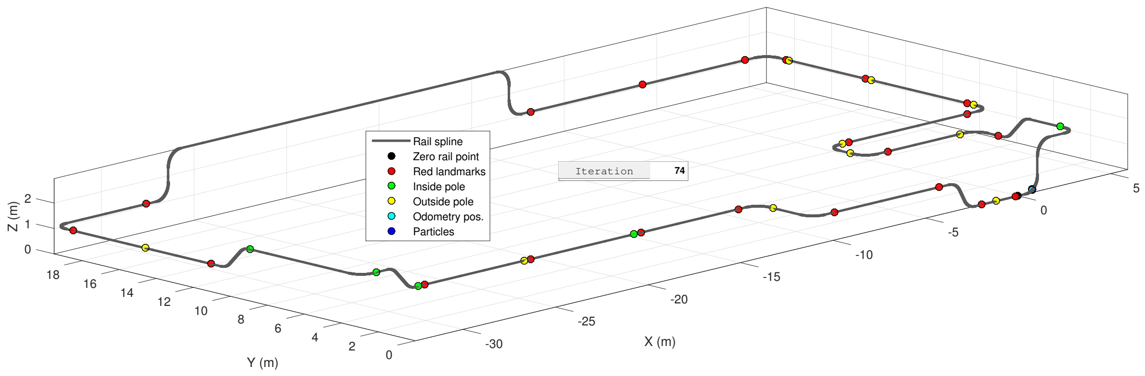
In the last example, the robot only moved forwards, which implies that the amount of displacement overestimated by the wheels in upward motion is compensated by the lack of displacement not accounted by the wheels in descending sections. As the robot returns back to the same initial point that is started to move, it is ex-



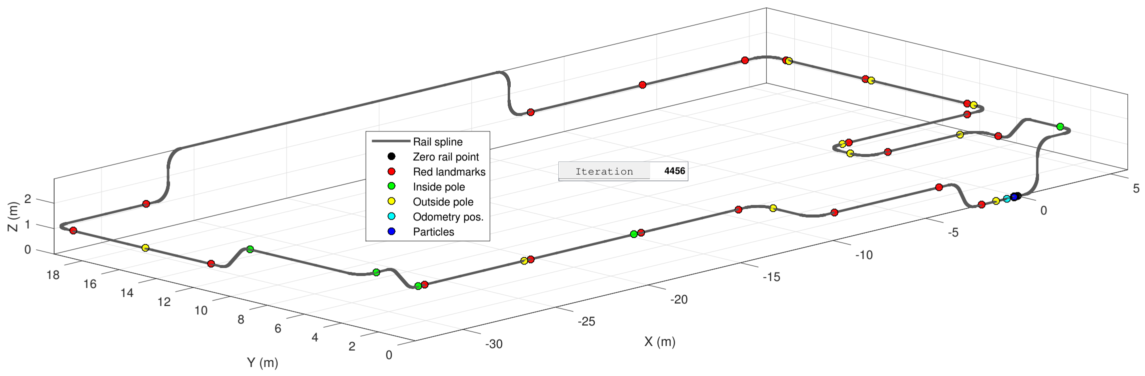
(a) Algorithm initialization with particles spread around the initial state.



(b) After a few iterations, the particles are concentrated in two hyposthesis.



(c) The ambiguity is solved when the robot reaches the first curve.



(d) Particles at the end of the test are near the true state.

Figure 4.1: **Simulation 1**: tracking problem in a complete loop test with 0.6m/s.

pected that the accumulated odometry error should be null. However, it was noted that the controllers are in more trouble to brake the robot than to lift it, hence the -59cm negative error. Furthermore, there are errors related to horizontal curved motion, and uncertainty in the rail length.

In this second simulation, the robot is forced to climb the first level transition of the rail and return to the base 5 times before going forwards (on the CCW direction) to complete a loop. In this way, the odometry error related to the vertical sections are not compensated, but accumulated, as the direction of motion changes.

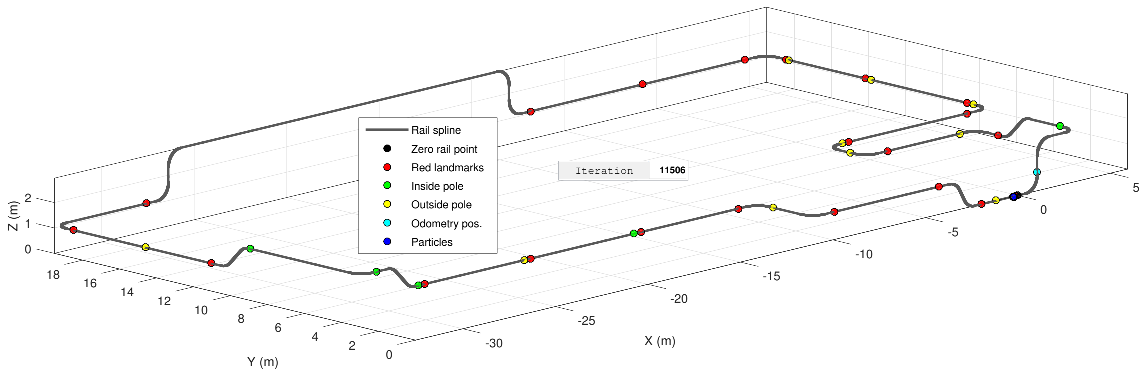


Figure 4.2: **Simulation 2**: tracking with accumulated odometry error.

After 5 times travelling through the up and down initial trajectory, the odometry error is of 1.28m, while the particles have successfully localized the robot with an error of just 15cm. At the end of the test, the odometry error is increased to 1.51m, while the estimation provided by the particle filter kept the error smaller than 25cm (Figure 4.2).

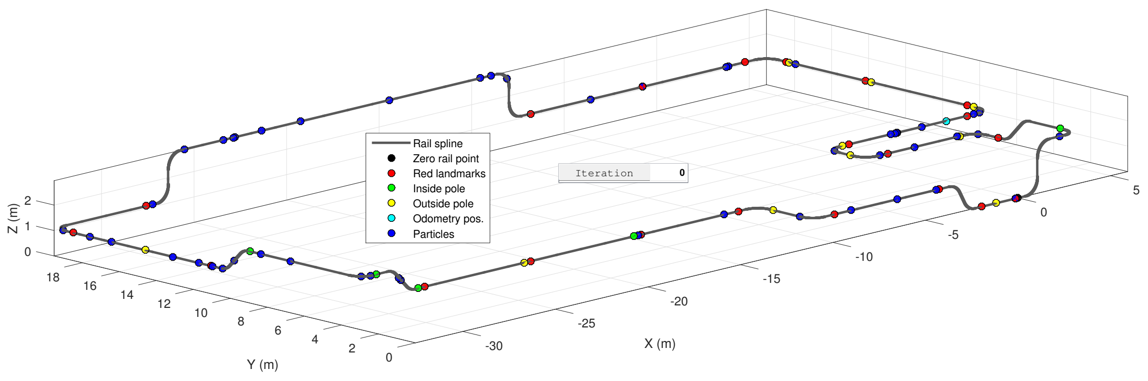
As the algorithm has some randomness included, the simulations were repeated several times. Moreover, various field tests data with different robot motion were considered in the simulations, which consistently yielded similar results. These simulations proved that the particle filter algorithm performs better than the simple odometric system for tracking problems.

**Simulation 3: global localization with 50 particles**

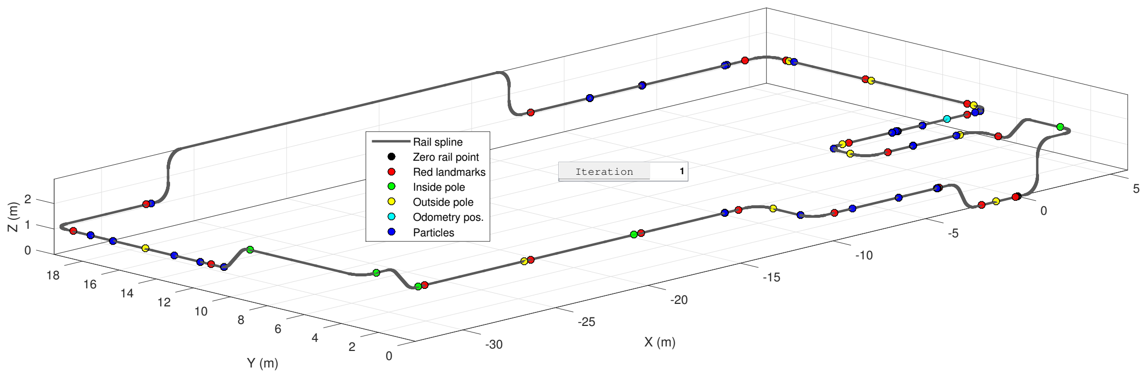
In order to test the particle filter capabilities, the algorithm was applied to solve a global localization problem, which is more complex than tracking. In this problem, the robot is completely unaware of its initial position in the environment.

The simulation was purposely initialized when DORIS was on a straight section of the rail at a height of 2.8m, as these characteristics are found in several regions of the rail, and, thus, the robot has multiple valid hypothesis of where it is. Fifty particles were randomly drawn throughout the rail path, as shown in Figure 4.3(a).

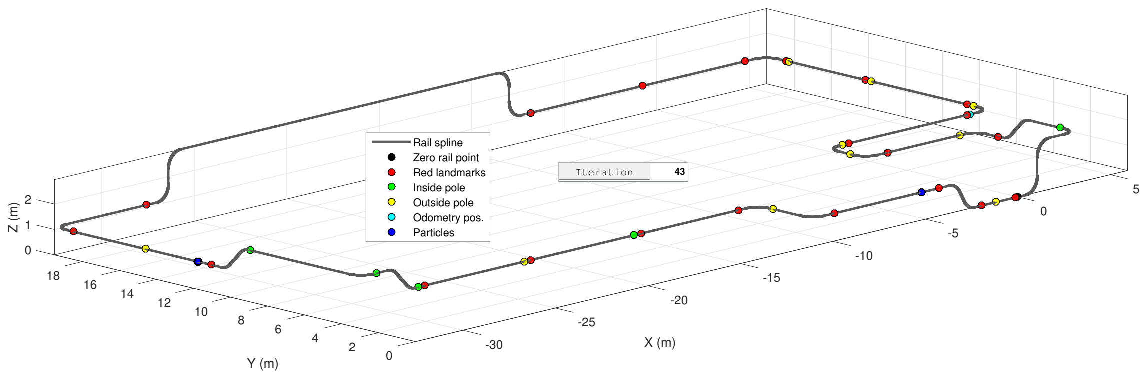
The cyan dot in Figure 4.3 represents the robot position estimated by odometry, which in the case of this test, indicates the true position of the robot with an



(a) Algorithm initialization with 50 particles randomly distributed.



(b) After a few iterations, only the particles at 2.8m survive.



(c) The algorithm has failed to estimate the true state.

Figure 4.3: **Simulation 3**: global localization with 50 particles.

error near 0.5m, as the odometric error was not accumulated in this case. Note in Figure 4.3(a) that no particles were assigned near the ground truth position.

After a single iteration of the algorithm, almost all the particles that do not match the real measurements taken by the robot are eliminated (Figure 4.3(b)). These particles are the ones that are at a height different than 2.8m, or in a curved part of the rail, or those that are observing landmarks, while the robot is not.

In Figure 4.3(c), the robot detects a red landmark, and only the particles that are near red markers survive. However, as there was no particle near the true state, the two surviving hypothesis are wrong and the robot gets lost. Eventually, all particles will be concentrated around one of the two wrong hypothesis, and the correct position of the robot will never be estimated.

This example shows the limitations of the particle filter presented in Section 2.5.3. In this case, the algorithm was not able to globally localize the robot because the posterior belief was coarsely estimated by a proposal density with a relatively small number of random particles. Abstractedly thinking of the rail as a histogram with 50 bins, each bin has 2.6m of rail length and each of the 50 particles should cover the features that are present in a 2.6m length of the rail. Even if one or a few particles happen to be initialized near the true state by random, they still have to survive the resampling steps against all the other particles, besides of having higher importance weights.

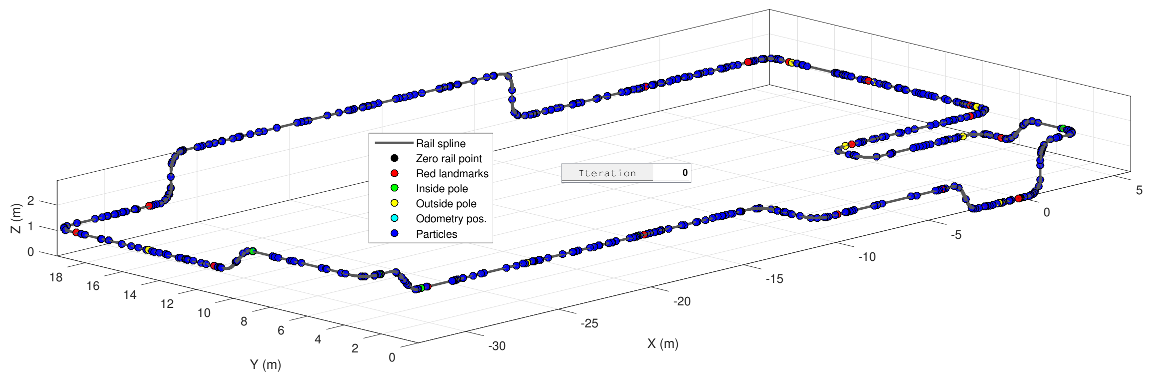
**Simulation 4: global localization with 500 particles**

One alternative to improve the algorithm in the global localization problem is to increase the number of particles. In this simulation, $N$ is increased to 500, while the remaining parameters and test data are the same as in Simulation 3. The results are shown in Figure 4.4.

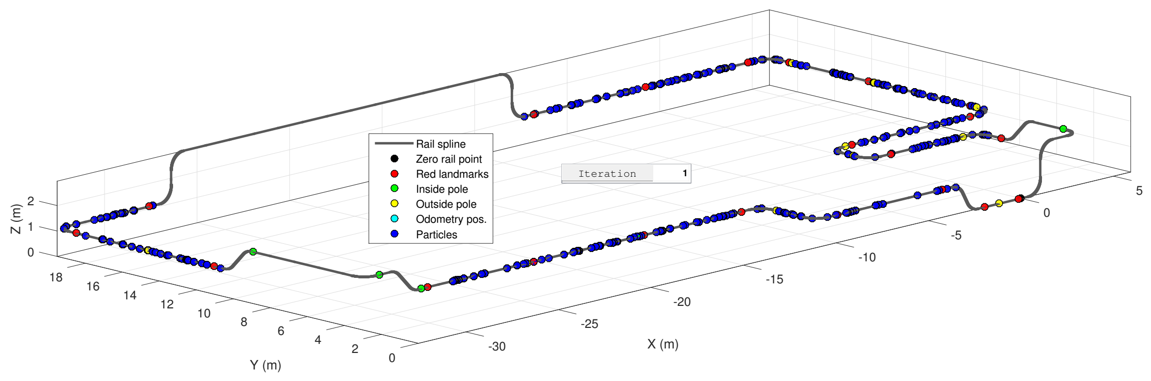
In Figure 4.4(a), the 500 particles have their positions randomly assigned in the initialization step, and now represent the rail states much better than the 50 particles of the previous example. A single step is enough to eliminate all the particles in positions with incompatible features than the straight rail at 2.8m height with no landmarks near it (Figure 4.4(b)).

When the robot moves forward and observes a red landmark, the particles near the red markers distributed on the rail receive high weights and are now concentrated in 9 different hypothesis (Figure 4.4(c)). Finally, when the robot turns left, it senses the turning by observing the rail geometry from the laser scanner, and only one of the 9 hypothesis (the correct one) has consistent features with the real measurements (Figure 4.4(d)). At this stage, all particles are concentrated near the true state and the robot has successfully self-localized after being lost in the beginning of the test. Running the simulation until the end of the test results in an estimation error of less than 25cm for the particle filter and near 60cm for the odometry system.

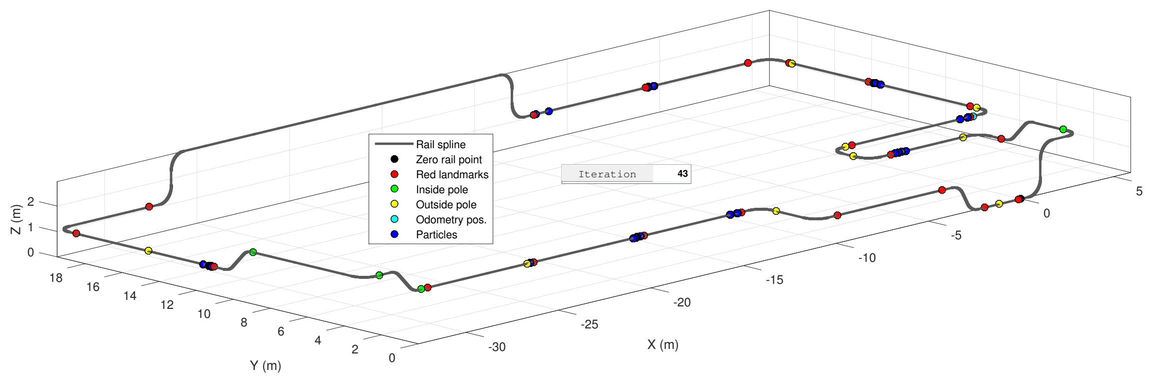
Despite of being able to solve the global localization problem for this specific simulation, the algorithm has to be repeatedly tested to prove its robustness and decrease the effect of randomness, inherent to a probabilistic filter, on the results. The simulation was repeated 10 times, but only 2 times the robot was successfully localized. If the number of particles used in the algorithm is increased, the algorithm would naturally perform better, as expected, as when the number of particles tends to infinity, the approximation of the posterior by the weighted samples drawn from a proposal distribution is exact.



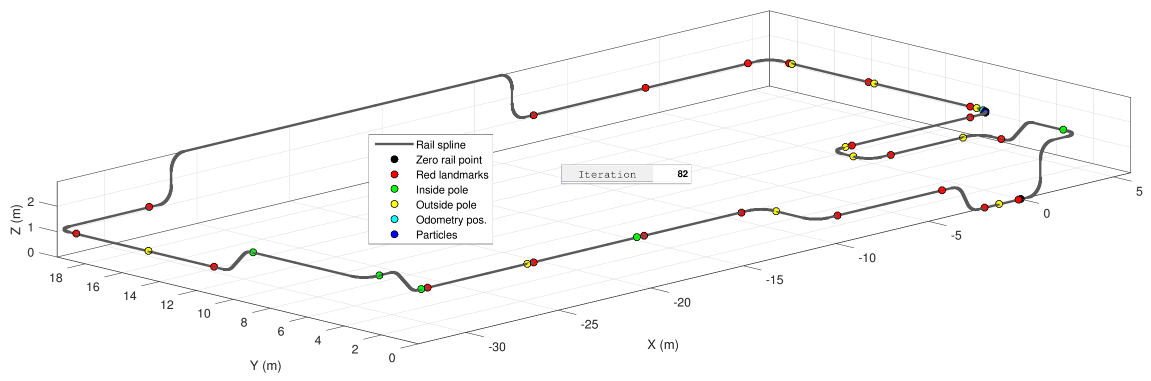
(a) Algorithm initialization with 500 particles randomly distributed.



(b) After one iteration, only the particles at 2.8m survive.



(c) After a red landmark detection, the particles are concentrated on 9 distinct regions.



(d) The true position is successfully estimated after the robot turns left.

Figure 4.4: **Simulation 4**: global localization with 500 particles.

Even though increasing the number of particles further more is an immediate solution, the also increasing computational costs may turn the algorithm infeasible to operate in real time. As an example, a simulation with 5000 particles in MATLAB environment took 2.5s in each iteration of the filter. Furthermore, the PF algorithm does not solve the kidnapped robot problem, as when all particles are concentrated near a single region, they will not jump to the true position, as already show above.

In order to deal with these problems, an extension of the particle filter is proposed in Section 4.2. The algorithm modifies the proposal distribution when the robot finds itself lost, resetting particles to positions that are consistent with the recent features observed by the robot.

## 4.2 History of Events Resetting (HER)

According to the simulation results presented in Section 4.1.1, the particle filter implementation performs well in tracking problems, but fails to solve the global localization and the kidnapped robot problems when the number of particles is small. In this condition, due to the particle filter limitations explained in Section 2.5.3, the algorithm can get ill-conditioned, which means that the robot might get completely lost forever.

Fortunately, as DORIS moves through a constrained path, it observes some characteristics of the environment in a predictable sequence. The *history of events* observed by the robot is, therefore, a valuable information that can be used to reset the robot position. When the robot finds itself lost, it can compare this sequence of events to a standard known sequence and reset particles in positions that comply with the observed history of events.

The selected events to be tracked by DORIS are the red landmark detection, the pole detection, and the curve types, as each of them is fairly recognizable and is a distinct feature of the environment. Red landmarks and poles can be detected by the algorithms described in Section 3.4.1 and Section 3.4.3, respectively, where a positive or negative detection is returned in the red landmark algorithm, and a detected pole type is returned in the pole detection algorithm.

Curve types can be recognized by analyzing turn rates measured by the IMU gyroscopes and the velocity provided by the motors drivers. A simple algorithm compares the pitch and yaw angular velocities to the sign of the linear velocity to identify one of the 5 types of tracks on the rail, which are left, right, up and down turns, and the straight track.

The robot keeps track of the detected events while it is moving through the rail on a single direction of motion. When it changes the motion direction and further detects one of these features, the event history is reset. This is because the standard events history is represented as the sequence of rail features on the counterclockwise direction. When the robot keeps a history of detected features, changes its motion direction, and continues to add events in this same sequence, it cannot be compared anymore to the standard sequence.

This approach is reasonable, as DORIS is intended to perform inspection rounds in the rail on a single direction of motion. Also, the robot may keep track of only the last 3 or 4 events, which are enough to make a sequence of events unique in the entire rail. Another assumption of the proposed algorithm is that the robot may detect only one event at a time. As this method is based on a sequence of detection to be compared to a standard sequence, if the robot detects two or more events at the same time, it will not be able to sort the correct order of detection, and the algorithm might fail.

The method proposed here resembles the *Sensor Resetting Localization* technique presented in Lenser & Veloso (2000), with the important difference that not only the most recent measurement, but the history of measurements is accounted to reset particles. The *History of Events Resetting* (HER) algorithm is depicted in Algorithm 15.

**Algorithm 15** History of Events Resetting (HER)

```
Require: $\mathcal{X}_t$, $\overline{\mathcal{W}}_t$, $E_t$, $z_t$
Ensure: $\mathcal{X}_t$, $\mathcal{W}_t$
 1: $N_{new} = \left(1 - \text{mean}(\overline{\mathcal{W}}_t)/w_{\text{thres}}\right) N$
 2: if $N_{new} > 0$ then
 3:     Search for the $K$ positions $s^k$ that comply with $E_t$
 4:     for $k = 1$ to $K$ do
 5:         for $i = 1$ to $N_{new}/K$ do
 6:             $x_t^i \sim p(z_t \mid s^k, m)$
 7:             $w_t^i = p(z_t \mid s^k, m)$
 8:             Replace sample with the lowest weight in $\mathcal{X}_t$ by $x_t^i$
 9:             Replace the corresponding weight in $\overline{\mathcal{W}}_t$ by $w_t^i$
10:         end for
11:     end for
12: end if
13: $\mathcal{W}_t = \overline{\mathcal{W}}_t$ normalized
14: return $\mathcal{X}_t$, $\mathcal{W}_t$
```

In line 1, the value $(1 - \text{mean}(\overline{\mathcal{W}}_t)/w_{\text{thres}})$ is calculated and can be understood as the probability of the robot being lost. $\overline{\mathcal{W}}_t$ in this case is the set of weights computed in the current iteration, not the weights $\mathcal{W}_t$ calculated recursively.

As in Algorithm 5, a number of new particles $N_{new}$ is computed as the percentage $(1 - \text{mean}(\overline{\mathcal{W}}_t)/w_{\text{thres}})$ of the total particles $N$. If the mean of the weights exceeds a threshold, then the robot is reasonably aware of its location and no particles should be reset. Otherwise, $N_{new}$ particles will be replaced in positions $s^k$ that comply with the recent history of events $E_t$.

The new particles are sampled from the distribution $p(z_t \mid s^k, m)$ in line 6 and the corresponding weights are computed according to $p(z_t \mid s^k, m)$ as well in line 7. The new samples replace particles in the set $\mathcal{X}_t$ that have the smallest weights. The importance factors are also replaced accordingly. Finally, after all the iterations, the new set of weights is normalized in line 13.

Mathematically speaking, the algorithm modifies the proposal distribution of the particle filter for a fraction of the particle set, as in the methods presented in Section 2.5.3. The modified proposal is given by the probability density of measuring a feature in a certain position given that this position should be compatible with the history of events observed by the robot. As the positions of the natural and artificial landmarks on the rail map are known, it is easy to sample particles from this distribution.

There exists the possibility of an incorrect detection of an event or the absence of detection, as for example red landmarks that are receiving too much light, turning the camera image bright enough to prevent its detection. Also, if two events are detected at the same time, the algorithm might get troubled to sort the correct order of events in the standard sequence. In these cases, the sequence of events may not be found in the standard array, and the algorithm considers only the most recent event in the history to perform particle resetting. In a worse situation, the incorrect sequence of detected events may match a sequence in the standard array that is not compatible with the actual robot position. In these situations, the algorithm may reset particles in the wrong place. However, the robot will eventually find itself lost and reset particles on the correct positions after properly detecting the events in the correct sequence.

The HER algorithm is triggered after DORIS PF iteration (Algorithm  14) is computed and when a red landmark or a pole is detected, and the event is added to $E_t$. The two steps combined sums up the complete particle filter implementation for DORIS localization, and simulation results are presented in section (4.2.1).

### 4.2.1 Simulation Results

In this section, the global localization problem is reassessed, but now with the HER algorithm as an extension to DORIS particle filter. All the parameters used in the previous simulations (Table 4.1) are kept the same and the parameter $w_{\text{thres}}$ is set as 10% of the maximum measurement probability. This implies that if the robot is less than 10% sure of its state, then HER is performed.

The number of events to be stored in the history array is set as 3. Three events in sequence were found to be unique, or, at most, have only two hypothesis in the standard sequence of events. In theory, a greater value would turn the sequences unique in the standard history, and particles would be reset without ambiguity. However, a small number of traced events diminishes the effects of an occasional incorrect detection of an event, as explained in Section 4.2.

**Simulations 5 and 6: HER and global localization with 50 particles**

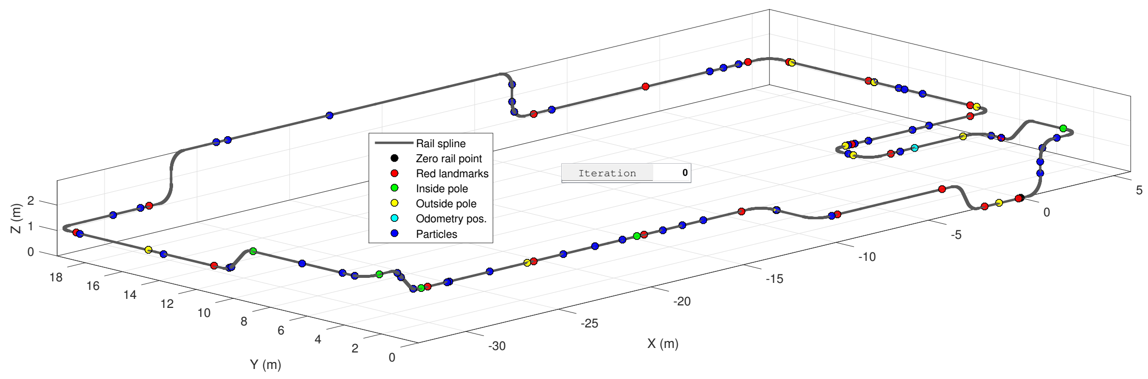
Simulation 5 considers 50 particles and the system is initialized with the robot having travelled 13.5m from the rail zero point. This is a similar initial position as the ones considered in Simulations 3 and 4 (straight section, 2.8m, no landmarks). The robot moves on the CCW direction with velocity 0.6m/s and the simulation results are shown in Figure 4.5.

The 50 particles are initialized at random, and no particle is in the vicinity of the actual position (Figure 4.5(a)). After iterations, all the particles are concentrated near an incorrect position (Figure 4.5(b)), and the robot gets lost. Note that, at this stage, a red landmark detection is already accounted in the history of events. In Figure 4.5(c), after the robot moved forwards and stored in the history array the detection of a red landmark followed by a right turn and an outside pole, the HER algorithm is triggered and resets particles on only the two regions of the rail that are consistent with this given sequence. The ambiguity is finally solved when DORIS turns right again, as in the wrong hypothesis the robot should turn left.

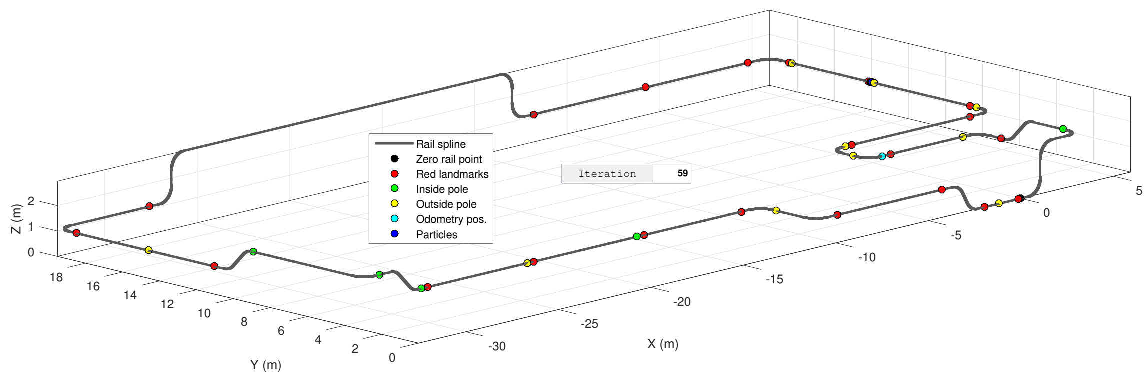
In Simulation 6, the algorithm was tested again using a different initial condition to prove its robustness and efficiency. The simulation results are shown in Figure 4.6. After the 50 particles were randomly initialized in the rail (Figure 4.6(a)) and were concentrated near a wrong hypothesis (Figure 4.6(b)), the robot was able to recover its position estimation correctly when the particles were reset (Figure 4.6(c)) by HER. In this case, DORIS sensed a red landmark, an up turn, a down turn, and an inside pole since the start of the simulation. Only the three most recent events are considered, and, as the detected sequence is unique in the standard events history, all particles were reset near the true state after DORIS detected an inside pole.

These simulations show that the robot position was correctly estimated in a few steps of the algorithm in a global localization problem. After DORIS moved and sensed three rail features, the particles converged to a region near the true state, even if they were randomly distributed over the rail environment in the initialization step. After running the simulations until DORIS reached the final position in the test, the estimation error was verified to be less than 25cm.

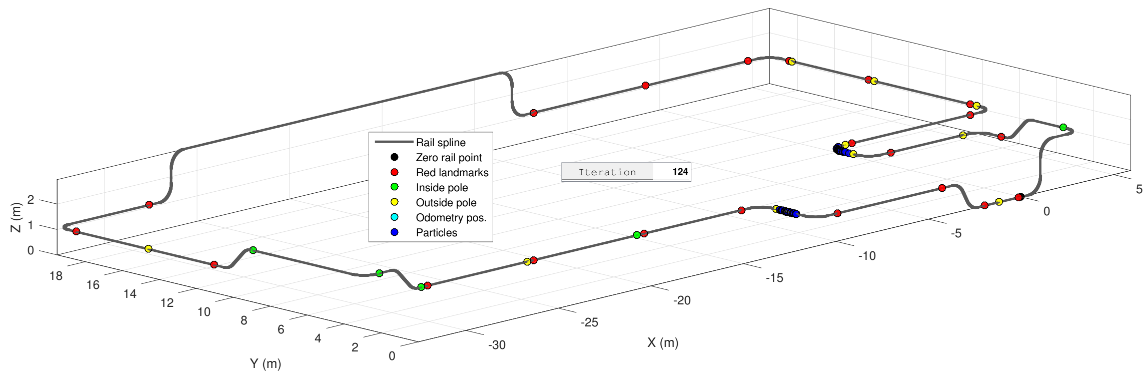
Figures 4.5 and 4.6 also show that the algorithm was able to solve the kidnapped robot problem. In Figs. 4.6(b) and 4.5(b), all particles converged to one single incorrect hypothesis, making the robot firmly believe it knows where it is while it



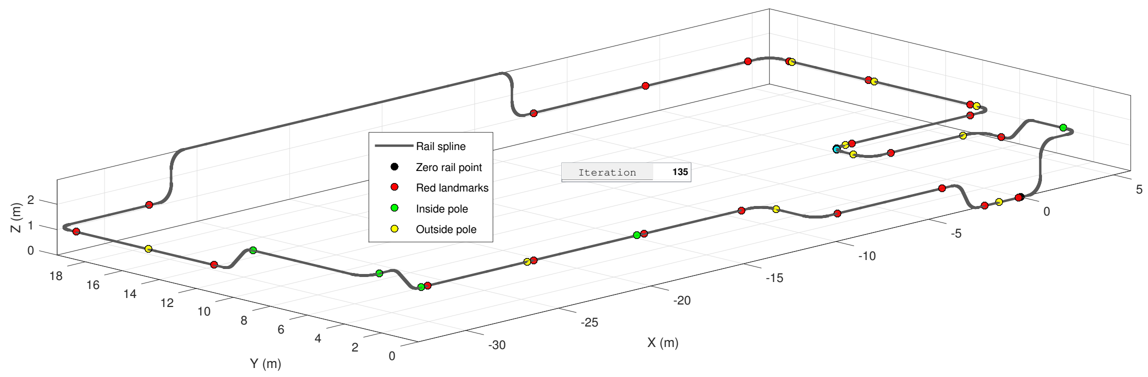
(a) Algorithm initialization with 50 particles randomly distributed.



(b) After some iterations, only an incorrect hypothesis survives.

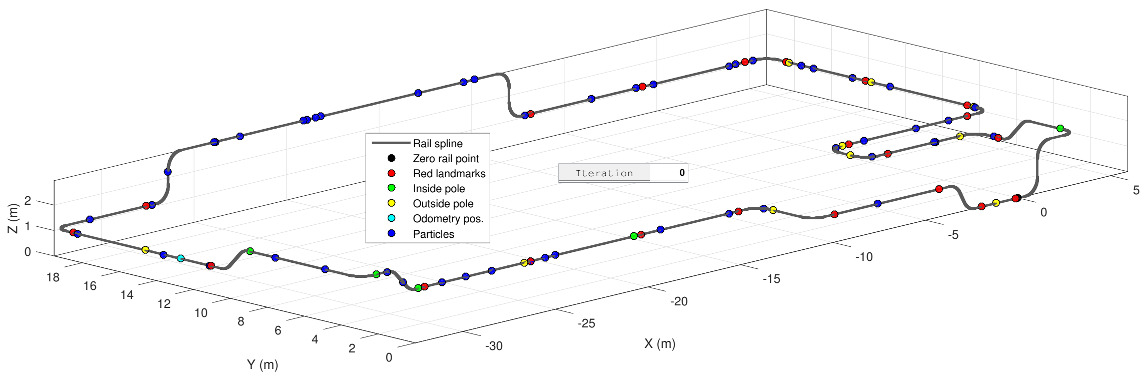


(c) After a pole detection, HER is called and particles are reset as two different hypothesis.

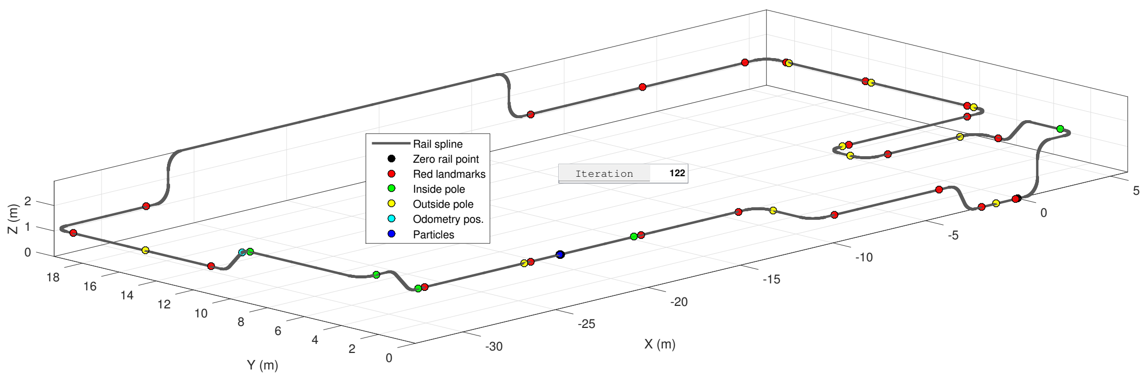


(d) The ambiguity is solved after the robot turns right again.

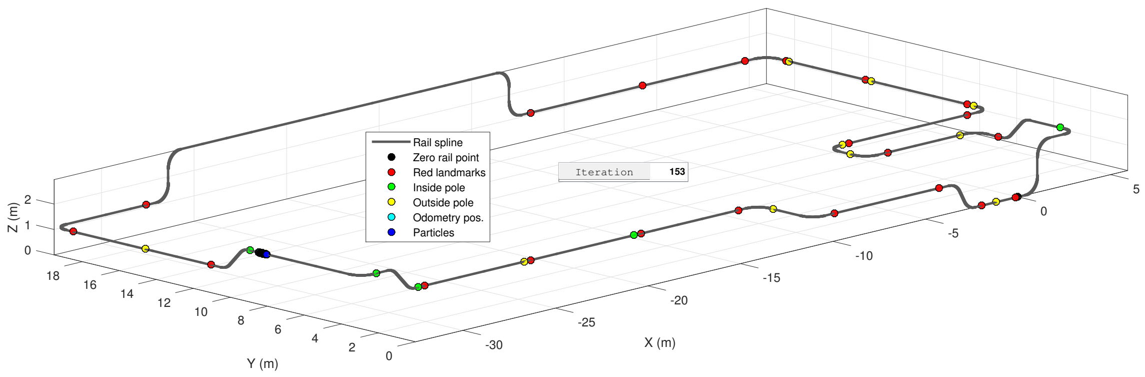
Figure 4.5: **Simulation 5**: global localization using HER with 50 particles.



(a) Algorithm initialization with 50 particles randomly distributed.



(b) After some iterations, only an incorrect hypothesis survives.



(c) After a pole detection, HER resets particles near the true position.

Figure 4.6: **Simulation 6**: global localization using HER with 50 particles.

does not. However, by analyzing the particle weights relative to an expected value, the robot noticed that it was lost and reset the particles to consistent assumptions about the true position with HER after it detected a sequence of events.

Several simulations were made to confirm the robustness of the algorithm, which was able to correctly localize the robot in the majority of the simulations, considering different initial conditions and tests data. However, the algorithm failed in a few simulations due to a violation of an assumption of the HER algorithm cited in Section 4.2. Unfortunately, the red landmarks were positioned on the rail without concerns about the chances of the robot detecting two events at the same time.

Unluckily, in some cases the red markers were placed with a distance to a pole exactly equal to the distance between the camera and the laser scanner on the robot, leading to a dual detection.

At first, this issue could probably be overcome by treating dual detection of events as a different event, but this also could lead to failures on the detection of the events. A possible solution would be to disconsider one of the features being tracked as events, as the red landmarks. Simulations done without these features in the history of events were successful in all cases, but with a slower convergence, as the robot had to move further to detect valid events. A more sound and simple solution would be to reposition the red markers to avoid dual detection with the poles, but new data would need to be acquired in the field, and this was left as a future work.

Other limitations of the algorithm are the ones already mentioned in Section 4.2, which are the chances of a misdetection (false positive or false negative) of an event, corrupting the correct event history and maybe the localization estimation. However, the algorithm would reliably recover the true state when a consistent detection of events is provided and the robot realizes it is lost. Once the particles populate the region near the actual position of the robot, their individual weights are considerably high, and the robot is correctly safe about its position. In this case, the standard particle filter will be able to track the robot position, the HER algorithm should not be called anymore, and the particles will remain near the true state.

# Chapter 5

# Conclusions

This dissertation has presented a localization algorithm proposed for DORIS, an autonomous rail-guided robot developed to operate in offshore platforms. Autonomous navigation systems in mobile robots must be resilient to failures and deal with uncertainty, so that, even though the characteristic rail motion of DORIS alleviates navigation problems, some localization issues still have to be addressed.

Probabilistic techniques are great in dealing with uncertainty and have received much interest in robotics recently. Therefore, its main algorithms applied to mobile robot localization were thoroughly investigated in the literature review of this text (Chapter 2). The characteristics of the algorithms, along with a detailed discussion about their advantages and limitations, were analyzed to conclude that a particle filter is the most appropriate method to solve the DORIS localization problem.

In Chapter 3, the robot environment and probabilistic models were presented. A rail was installed in a utility plant in CENPES to test the robot and demonstrate its functionalities. The environment characteristics, such as the rail Cartesian position, and its natural and artificial landmarks positions, were considered as known and included in a map. As the rail track is a unidimensional element, all these features can be associated to a single parameter that represents the travelled distance from the zero point position of the rail. Therefore, it suffices to estimate this single parameter to provide the robot localization.

DORIS motion model is trivially defined as the displacement provided by the velocity in a time interval with a given odometric uncertainty, and different sensors were considered in the perception model. While an IMU provides the robot pitch angle, a laser scanner and a camera detects natural and artificial landmarks.

In the basic implementation of the particle filter, the proposal distribution is given by the motion model and is used to sample a set of particles, which are weighted according to the perception model to approach the posterior belief. The resampling step is performed whenever the effective number of particles is too low.

The simulation results shown in Chapter 4 using data acquired in field tests with DORIS confirm that the particle filter performs better than a simple odometry system in tracking problems, that is, when the robot initial condition is partially known. However, the particle filter was not able to correctly estimate the robot position when its initial state was completely unknown (global localization problem).

A novel extension to the basic particle filter was then proposed. The *History of Events Resetting* algorithm modifies the proposal distribution for a number of particles calculated proportionally to how lost the robot is. A new set of particles is built after resetting some of them to positions that are consistent with the recent feature observations made by the robot.

The results showed that the extended particle filter with HER algorithm was able to solve the global localization and even the kidnapped robot problems using a very small number of particles compared to what is used in other particle filter implementations. In the global localization problem, while only 50 particles converged to the true state after the robot detected up to 3 rail features, a number 100 or even 1000 times greater would be necessary to reliably localize the robot using the standard particle filter algorithm. Moreover, the HER algorithm was able to solve the kidnapped robot problem, while the standard PF did not.

This demonstrates that the algorithm proposed in this work provides a more reliable and efficient solution to a rail-guided robot, and specifically for DORIS. Furthermore, it is computationally more efficient and can be easily implemented in a real time application.

The limitations of the proposed method are related to the assumption of a completely known map, and to the rail features recognition models. The red landmarks can have false negative detections if the image is very bright due to light conditions, and also, the markers can wear over time due to weather conditions. In addition, the poles may have false negative detections if the robot is moving too fast, for example. Furthermore, the distance from the robot to the floor measured by the laser scanner can be corrupted by equipments positioned under the rail or people following the robot, although neither of them should occur due to safety reasons. Finally, the HER algorithm considers the detection of only one event at a time, being unable to deal with dual event detection.

## 5.1 Future Work

In the following, suggestions of future work are presented in order to improve the developed algorithm and extend it to more general implementations:

- **Include other measurements in the system**, as the heading angle. Despite of being a challenge to use IMU heading data in such an adverse environment with significant magnetic interference, the heading information would be great to distinguish localization hypothesis that are not discriminate in the present implementation.

- **Reposition the red landmarks** on the rail and acquire new field tests data. The correct positioning of the red markers would avoid the dual event detection problem, described in Section 4.2.1.

- **Change the type of artificial landmarks** to a more reliable one. Vision landmarks can wear over time, are subject to light conditions, and false negative detections may occur. Possibly, another type of landmark positioned on the rail, as a magnet (and a hall sensor in the robot), would be detected with more robustness.

- **Include more rail fixations in the map**. In this work, only the poles fixed to the floor were considered, as they are easily recognizable. With a more elaborated detection model, other types of rail fixations could be included in the map as landmarks.

- **Implement and validate the algorithm in real time**, that is, integrate the code in DORIS autonomous navigation system and test it in the field. The filter parameters should also be tuned. Usually, switching from the comfortable simulation environment to a real time implementation requires extensive work.

- **Relax the assumptions considered in the robot models**. To derive the map, motion, and perception models in Chapter 3, assumptions A1 (zero roll) and A2 (only one curve at a time) were considered. These hypothesis have to be relaxed or disregarded to derive universal models for any type of rail.

- **Extend the rail map to have more than one track**. The use of DORIS in larger environments might need a rail network with more than one open or closed circuit. In this case, the algorithm should consider modifications such as a local and global map, switch modelling, and inclusion of a track ID as a state variable, as in García (2012).

- **Simultaneous localization and mapping**. The SLAM problem was not considered in this work, as the map was supposed to be known. However, this is not entirely true, as there may have been measurement errors of the rail segments, and the rail optimization to close the loop, described in Section 3.1.1, considers approximations. Moreover, for longer rails, taking manual measurements of each segment may be a costly task, and the rail path can always be changed if needed, requiring new measurements. The automatic update of the map in SLAM could deal with all these issues.

# Bibliography

Advanced Navigation (2013), *Spatial OEM reference manual*, 2.7 edn.

Aghamohammadi, A. A., Taghirad, H. D., Tamjidi, A. H. & Mihankhah, E. (2007), Feature-based laser scan matching for accurate and high speed mobile robot localization., *in* 'EMCR', Citeseer.

Angermann, M. & Robertson, P. (2012), 'Footslam: Pedestrian simultaneous localization and mapping without exteroceptive sensors—hitchhiking on human perception and cognition', *Proceedings of the IEEE* **100**(Special Centennial Issue), 1840–1848.

Anisi, D. A., Persson, E. & Heyer, C. (2011), Real-world demonstration of sensor-based robotic automation in oil & gas facilities, *in* 'Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on', IEEE, pp. 235–240.

Anisi, D. A., Skourup, C. & Petrochemicals, A. (2012), A step-wise approach to oil and gas robotics, *in* 'IFAC Workshop on Automatic Control in Offshore Oil and Gas Production, Trondheim, Norway', Vol. 31.

Anisi, D., Gunnar, J., Lillehagen, T. & Skourup, C. (2010), Robot automation in oil and gas facilities: Indoor and onsite demonstrations, *in* 'Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)', pp. 4729–4734.

Bailey, T. & Durrant-Whyte, H. (2006), 'Simultaneous localization and mapping (slam): Part ii', *IEEE Robotics & Automation Magazine* **13**(3), 108–117.

Bar-Shalom, Y., Li, X. R. & Kirubarajan, T. (2004), *Estimation with applications to tracking and navigation: theory algorithms and software*, John Wiley & Sons.

Bengel, M. & Pfeiffer, K. (2007), Mimroex mobile maintenance and inspection robot for process plants. *Fraunhofer Institute for Manufacturing Engineering and Automation IPA*, pp. 1–2.

Bengel, M., Pfeiffer, K., Graf, B., Bubeck, A. & Verl, A. (2009), Mobile robots for offshore inspection and manipulation, *in* 'Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS). on', pp. 3317–3322.

Besl, P. J. & McKay, N. D. (1992), Method for registration of 3-d shapes, *in* 'Robotics-DL tentative', International Society for Optics and Photonics, pp. 586–606.

Biswas, J. & Veloso, M. (2010), Wifi localization and navigation for autonomous indoor mobile robots, *in* 'Robotics and Automation (ICRA), 2010 IEEE International Conference on', IEEE, pp. 4379–4384.

Bjerkeng, M., Transeth, A. A., Pettersen, K. Y., Kyrkjebo, E. & Fjerdingen, S. A. (2011), Active camera control with obstacle avoidance for remote operations with industrial manipulators: Implementation and experimental results, *in* 'Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on', IEEE, pp. 247–254.

Bolić, M., Djurić, P. M. & Hong, S. (2003), New resampling algorithms for particle filters, *in* 'Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on', Vol. 2, IEEE, pp. II–589.

Bonin-Font, F., Ortiz, A. & Oliver, G. (2008), 'Visual navigation for mobile robots: A survey', *Journal of intelligent and robotic systems* **53**(3), 263–296.

Borgerink, D., Stegenga, J., Brouwer, D., Wortche, H. & Stramigioli, S. (2014), Rail-guided robotic end-effector position error due to rail compliance and ship motion, *in* 'Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on', IEEE, pp. 3463–3468.

Bos, B. v. d., Mallion, A., Wilson, C., Zwicker, E., Schler, A., Black, T. et al. (2015), Robotic inspection solutions for petrochemical pressure vessels, *in* 'Abu Dhabi International Petroleum Exhibition and Conference', Society of Petroleum Engineers.

Carvalho, F., Raposo, A., Santos, I. & Galassi, M. (2014), Virtual reality techniques for planning the offshore robotizing, *in* 'Industrial Informatics (INDIN), 2014 12th IEEE International Conference on', IEEE, pp. 353–358.

Carvalho, G., Freitas, G., Costa, R., Carvalho, G., Oliveira, J., Netto, S., Silva, E., Xaud, M., Hsu, L., Motta-Ribeiro, G. et al. (2013), Doris-monitoring robot for offshore facilities, *in* 'Offshore Technology Conference Brasil'.

Chen, Z. (2003), 'Bayesian filtering: From kalman filters to particle filters, and beyond', *Statistics* **182**(1), 1–69.

Chitta, S., Vernaza, P., Geykhman, R. & Lee, D. D. (2007), Proprioceptive localilzatilon for a quadrupedal robot on known terrain, *in* 'Robotics and Automation, 2007 IEEE International Conference on', IEEE, pp. 4582–4587.

Christensen, L., Fischer, N., Kroffke, S., Lemburg, J. & Ahlers, R. (2011), 'Cost-effective autonomous robots for ballast water tank inspection', *Journal of ship production and design* **27**(3), 127–136.

Christensen, L., Kirchner, F., Fischer, N., Ahlers, R., Psarros, G. & Etzold, L. (2011), Tank inspection by cost effective rail based robots, *in* 'Proceedings of International Conference on Computer Applications in Shipbuilding (ICCAS)'.

Coppersmith, D. & Winograd, S. (1987), Matrix multiplication via arithmetic progressions, *in* 'Proceedings of the nineteenth annual ACM symposium on Theory of computing', ACM, pp. 1–6.

Crisan, D. & Obanubi, O. (2012), 'Particle filters with random resampling times', *Stochastic processes and their applications* **122**(4), 1332–1368.

De Cecco, M. (2003), 'Sensor fusion of inertial-odometric navigation as a function of the actual manoeuvres of autonomous guided vehicles', *Measurement Science and Technology* **14**(5), 643.

Diosi, A. & Kleeman, L. (2005), Laser scan matching in polar coordinates with application to slam, *in* 'Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on', IEEE, pp. 3317–3322.

Doh, N. L., Choset, H. & Chung, W. K. (2006), 'Relative localization using path odometry information', *Autonomous Robots* **21**(2), 143–154.

Douc, R. & Cappé, O. (2005), Comparison of resampling schemes for particle filtering, *in* 'Image and Signal Processing and Analysis, 2005. ISPA 2005. Proceedings of the 4th International Symposium on', IEEE, pp. 64–69.

Doucet, A., Godsill, S. & Andrieu, C. (2000), 'On sequential monte carlo sampling methods for bayesian filtering', *Statistics and computing* **10**(3), 197–208.

Durrant-Whyte, H. & Bailey, T. (2006), 'Simultaneous localization and mapping: part i', *Robotics & Automation Magazine, IEEE* **13**(2), 99–110.

Engelson, S. P. & McDermott, D. V. (1992), Error correction in mobile robot map learning, *in* 'Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on', IEEE, pp. 2555–2560.

Faber Archila, J. & Becker, M. (2013), Study of robots to pipelines, mathematical models and simulation, *in* 'Robotics Symposium and Competition (LARS/LARC), 2013 Latin American', IEEE, pp. 18–23.

Faria, R. O., Kucharczak, F., Freitas, G. M., Leite, A. C., Lizarralde, F., Galassi, M. & From, P. J. (2015), A methodology for autonomous robotic manipulation of valves using visual sensing, Vol. 48, Elsevier, pp. 221–228.

Fjerdingen, S. A., Bjerkeng, M., Transeth, A. A., Kyrkjebo, E. & Royroy, A. (2012), A learning camera platform for remote operations with industrial manipulators, *in* 'Proc. of the 2012 IFAC Workshop on Automatic Control in Offshore Oil and Gas Production', pp. 39–46.

Fox, D. (2001), Kld-sampling: Adaptive particle filters, *in* 'Advances in neural information processing systems', pp. 713–720.

Fox, D. (2003), 'Adapting the sample size in particle filters through kld-sampling', *The international Journal of robotics research* **22**(12), 985–1003.

Fox, D., Burgard, W., Dellaert, F. & Thrun, S. (1999), 'Monte carlo localization: Efficient position estimation for mobile robots', *AAAI/IAAI* **1999**, 343–349.

Fox, D., Burgard, W., Kruppa, H. & Thrun, S. (2000), 'A probabilistic approach to collaborative multi-robot localization', *Autonomous robots* **8**(3), 325–344.

Fox, D., Burgard, W. & Thrun, S. (1998), 'Active markov localization for mobile robots', *Robotics and Autonomous Systems* **25**(3), 195–207.

Fox, D., Thrun, S., Burgard, W. & Dellaert, F. (2001), Particle filters for mobile robot localization, *in* 'Sequential Monte Carlo methods in practice', Springer, pp. 401–428.

Freitas, R. S. (2016), Arquitetura híbrida e controle de missão de robôs autônomos, Master's thesis, Federal University of Rio de Janeiro.

Freitas, R. S., Xaud, M. F., Marcovistz, I., Neves, A. F., Faria, R. O., Carvalho, G. P., Hsu, L., Nunes, E. V., Peixoto, A. J., Lizarralde, F. et al. (2015), The embedded electronics and software of doris offshore robot, Vol. 48, Elsevier, pp. 208–213.

From, P. (2010), Off-Shore Robotics: Robust and Optimal Solutions for Autonomous Operation, PhD thesis, Norwegian University of Science and Technology.

Galassi, M., Røyrøy, A., Carvalho, G. P. S., Freitas, G. M., From, P. J., Costa, R. R., Lizarralde, F., Hsu, L., de Carvalho, G. H., de Oliveira, J. F. et al. (2014), Doris-a mobile robot for inspection and monitoring of offshore facilities, *in* 'Anais do XX Congresso Brasileiro de Automática'.

García, A. C. (2012), Railslam: Simultaneous localization and mapping for railways, Master's thesis, University of Oviedo.

Goldberg, D. E. et al. (1989), *Genetic algorithms in search optimization and machine learning*, Vol. 412, Addison-wesley Reading Menlo Park.

Gordon, N. J., Salmond, D. J. & Smith, A. F. (1993), Novel approach to nonlinear/non-gaussian bayesian state estimation, *in* 'IEE Proceedings F (Radar and Signal Processing)', Vol. 140, IET, pp. 107–113.

Graf, B. & Pfeiffer, K. (2007), Mobile robots for offshore inspection and manipulation, *in* 'Proc. Int. Petroleum Technology Conf. (IPTC)'.

Graf, B. & Pfeiffer, K. (2008), Mobile robotics for offshore automation, *in* 'Proceedings of the EURON/IARP International Workshop on Robotics for Risky Interventions and Surveillance of the Environment, Benicassim, Spain'.

Hahnel, D., Burgard, W., Fox, D., Fishkin, K. & Philipose, M. (2004), Mapping and localization with rfid technology, *in* 'Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on', Vol. 1, IEEE, pp. 1015–1020.

Hasberg, C., Hensel, S. & Stiller, C. (2012), 'Simultaneous localization and mapping for path-constrained motion', *Intelligent Transportation Systems, IEEE Transactions on* **13**(2), 541–552.

Heirich, O., Robertson, P., Garcia, A. C. & Strang, T. (2012), Bayesian train localization method extended by 3d geometric railway track observations from inertial sensors, *in* 'Information Fusion (FUSION), 2012 15th International Conference on', IEEE, pp. 416–423.

Heirich, O., Robertson, P. & Strang, T. (2013), Railslam-localization of rail vehicles and mapping of geometric railway tracks, *in* 'Robotics and Automation (ICRA), 2013 IEEE International Conference on', IEEE, pp. 5212–5219.

Hensel, S. & Hasberg, C. (2010), Probabilistic landmark based localization of rail vehicles in topological maps, *in* 'Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on', IEEE, pp. 4824–4829.

Hensel, S., Hasberg, C. & Stiller, C. (2011), 'Probabilistic rail vehicle localization with eddy current sensors in topological maps', *Intelligent Transportation Systems, IEEE Transactions on* **12**(4), 1525–1536.

Hol, J. D., Schon, T. B. & Gustafsson, F. (2006), On resampling algorithms for particle filters, *in* 'Nonlinear Statistical Signal Processing Workshop, 2006 IEEE', IEEE, pp. 79–82.

Isard, M. & Blake, A. (1998), 'Condensation—conditional density propagation for visual tracking', *International journal of computer vision* **29**(1), 5–28.

Johnsen, S., Ask, R. & Roisli, R. (2007), Reducing risk in oil and gas production operations, *in* 'Critical infrastructure protection', Springer, pp. 83–95.

Julier, S. J. & Uhlmann, J. K. (1997), New extension of the kalman filter to nonlinear systems, *in* 'AeroSense'97', International Society for Optics and Photonics, pp. 182–193.

Kalman, R. E. (1960), 'A new approach to linear filtering and prediction problems', *Journal of Fluids Engineering* **82**(1), 35–45.

Kanazawa, K., Koller, D. & Russell, S. (1995), Stochastic simulation algorithms for dynamic probabilistic networks, *in* 'Proceedings of the Eleventh conference on Uncertainty in artificial intelligence', Morgan Kaufmann Publishers Inc., pp. 346–351.

Kwok, C., Fox, D. & Meila, M. (2003), Adaptive real-time particle filters for robot localization, *in* 'Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on', Vol. 2, IEEE, pp. 2836–2841.

Kydd, K., Macrez, S., Pourcel, P. et al. (2015), Autonomous robot for gas and oil sites, *in* 'SPE Offshore Europe Conference and Exhibition', Society of Petroleum Engineers.

Kyrkjebø, E., Liljebäck, P. & Transeth, A. (2009), A robotic concept for remote inspection and maintenance on oil platforms, *in* 'Proc. Int. Conf. on Ocean, Offshore and Arctic Engineering (OMAE)'.

Lenser, S. & Veloso, M. (2000), Sensor resetting localization for poorly modelled mobile robots, *in* 'Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on', Vol. 2, IEEE, pp. 1225–1232.

Leonard, J. J. & Durrant-Whyte, H. F. (1991), 'Mobile robot localization by tracking geometric beacons', *Robotics and Automation, IEEE Transactions on* **7**(3), 376–382.

Liu, J. S. (1996), 'Metropolized independent sampling with comparisons to rejection sampling and importance sampling', *Statistics and Computing* **6**(2), 113–119.

Liu, J. S. & Chen, R. (1998), 'Sequential monte carlo methods for dynamic systems', *Journal of the American statistical association* **93**(443), 1032–1044.

Metropolis, N. & Ulam, S. (1949), 'The monte carlo method', *Journal of the American statistical association* **44**(247), 335–341.

Montemerlo, M., Becker, J., Bhat, S., Dahlkamp, H., Dolgov, D., Ettinger, S., Haehnel, D., Hilden, T., Hoffmann, G., Huhnke, B. et al. (2008), 'Junior: The stanford entry in the urban challenge', *Journal of field Robotics* **25**(9), 569–597.

Montemerlo, M. & Thrun, S. (2007), 'Fastslam 2.0', *FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics* pp. 63–90.

Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B. et al. (2002), Fastslam: A factored solution to the simultaneous localization and mapping problem, *in* 'AAAI/IAAI', pp. 593–598.

Montermerlo, M. (2003), FastSLAM: a factored solution to the simultaneous localization and mapping problem with unknown data association, PhD thesis, PhD thesis, Carnegie Mellon University.

Ndjeng, A. N., Gruyer, D. & Glaser, S. (2008), Improvement of the proprioceptive-sensors based ekf and imm localization, *in* 'Intelligent Transportation Systems, 2008. ITSC 2008. 11th International IEEE Conference on', IEEE, pp. 900–905.

Nguyen, V., Martinelli, A., Tomatis, N. & Siegwart, R. (2005), A comparison of line extraction algorithms using 2d laser rangefinder for indoor mobile robotics, *in* 'Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on', IEEE, pp. 1929–1934.

Nieto, J., Bailey, T. & Nebot, E. (2007), 'Recursive scan-matching slam', *Robotics and Autonomous Systems* **55**(1), 39–49.

Nistér, D., Naroditsky, O. & Bergen, J. (2004), Visual odometry, *in* 'Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on', Vol. 1, IEEE, pp. I–652.

North, E., Georgy, J., Iqbal, U., Tarbochi, M. & Noureldin, A. (2012), 'Improved inertial/odometry/gps positioning of wheeled robots even in gps-denied environments', *Book Chapter in "Global Navigation Satellite Systems" In-Tech, ISBN* pp. 978–953.

NREC/CMU (2012), Sensabot: A safe and cost-effective inspection solution. *Journal of Petroleum Technology*, pp. 32-–34.

O'Kane, J. M. (2006), Global localization using odometry, *in* 'Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on', IEEE, pp. 37–42.

P. Liljebäck, T. K. & Schumann-Olsen, H. (2005), '"robotic technologies for an unmanned platform". sintef, report stf90 f05405'.

Pfeiffer, K., Bengel, M. & Bubeck, A. (2011), Offshore robotics-survey, implementation, outlook, *in* 'Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on', IEEE, pp. 241–246.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R. & Ng, A. (2009), ROS: an open-source robot operating system, *in* 'Proc. Int. Conf. on Robotics and Automation (ICRA), ser. Open-Source Software Workshop'.

Robertson, P., Frassl, M., Angermann, M., Doniec, M., Julian, B. J., Garcia Puyol, M., Khider, M., Lichtenstern, M. & Bruno, L. (2013), Simultaneous localization and mapping for pedestrians using distortions of the local magnetic field intensity in large indoor environments, *in* 'Indoor Positioning and Indoor Navigation (IPIN), 2013 International Conference on', IEEE, pp. 1–10.

Santos, I. H. F., Ribeiro, G. M., Coutinho, F., Hsu, L., Raposo, A., Carvalho, F., Medeiros, D., Galassi, M., Costa, R., From, P. A. et al. (2013), A robotics framework for planning the offshore robotizing using virtual reality techniques, *in* 'OTC Brasil', Offshore Technology Conference.

Shukla, A. & Karki, H. (2013), A review of robotics in onshore oil-gas industry, *in* 'Mechatronics and Automation (ICMA), 2013 IEEE International Conference on', IEEE, pp. 1153–1160.

SINTEF (2008), ‘Robots taking over the job on offshore oil drilling platforms’, ScienceDaily. Available: *Link* [Acessed in April 6, 2016].

Skourup, C. & Pretlove, J. (2009), ‘The robotized field operator’, *ABB Review* (1), 68–73.

Smith, A., Doucet, A., de Freitas, N. & Gordon, N. (2013), *Sequential Monte Carlo methods in practice*, Springer Science & Business Media.

Thrun, S. (1998), ‘Bayesian landmark learning for mobile robot localization’, *Machine Learning* **33**(1), 41–76.

Thrun, S. (2012), ‘Artificial intelligence for robotics’, Udacity online course CS373.

Thrun, S., Burgard, W. & Fox, D. (2005), *Probabilistic robotics*, MIT press.

Thrun, S., Fox, D., Burgard, W. & Dellaert, F. (2001), ‘Robust monte carlo localization for mobile robots’, *Artificial intelligence* **128**(1), 99–141.

Thrun, S., Fox, D., Burgard, W. et al. (2000), Monte carlo localization with mixture proposal distribution, *in* ‘AAAI/IAAI’, pp. 859–865.

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G. et al. (2007), Stanley: The robot that won the darpa grand challenge, *in* ‘The 2005 DARPA Grand Challenge’, Springer, pp. 1–43.

Transeth, A. A., Galassi, M., Royroy, A. K., Schumann-Olsen, H. et al. (2013), Robotics for the petroleum industry-challenges and opportunities, *in* ‘SPE Middle East Intelligent Energy Conference and Exhibition’, Society of Petroleum Engineers.

Transeth, A. A., Skotheim, O., Schumann-Olsen, H., Johansen, G., Thielemann, J. & Kyrkjebo, E. (2010), A robotic concept for remote maintenance operations: A robust 3d object detection and pose estimation method and a novel robot tool, *in* ‘Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on’, IEEE, pp. 5099–5106.

Van Der Merwe, R. (2004), Sigma-point Kalman filters for probabilistic inference in dynamic state-space models, PhD thesis, Oregon Health & Science University.

Vatland, S. & Svenes, M. (2008), ‘Tail integrated operations - project newsletter’, Available: *Link* [Acessed in April 6, 2016].

Xaud, M. F. d. S. (2016), Modeling, control and electromechanical design of a modular lightweight manipulator for interaction and inspection tasks, Master's thesis, Federal University of Rio de Janeiro.

Yuh, J. (2000), 'Design and control of autonomous underwater robots: A survey', *Autonomous Robots* **8**(1), 7–24.