Miguel Ângelo Ferreira Torres

Exploiting particle-filter based fusion in mobile robot localization

Dissertation submitted in partial fulfilment of the requirements for the degree of Master of Science in Electrical and Computer Engineering

Coimbra, September 2018

UNIVERSIDADE DE COIMBRA

University of Coimbra

Faculty of Sciences and Technology

Department of Electrical and Computer Engineering

Exploiting particle-filter based fusion in mobile robot localization

Miguel Ângelo Ferreira Torres

A dissertation presented for the degree of

Master of Science in Electrical and Computer Engineering

Coimbra, 2018

Exploiting particle-filter based fusion in mobile robot localization

Supervisor:

Prof. Doutor Urbano José Carreira Nunes

Co-Supervisors:

Master Luís Garrote

Prof. Doutor Cristiano Premebida

Jury:

Prof. Doutor Urbano José Carreira Nunes

Prof. Doutor Manuel Marques Crisóstomo

Prof.ª Doutora Ana Cristina Barata Pires Lopes

A dissertation submitted in partial satisfaction of the requirements for the degree of Master of Science in Electrical and Computer Engineering

Coimbra, 2018

Acknowledgements

This dissertation could not have been completed without the guidance and assistance of many people to whom I would like to express my appreciation and gratitude. I am very grateful to my advisors, Professor Doutor Urbano Nunes and Professor Doutor Cristiano Premebida for giving me the chance to work on a subject that I enjoy, for offering me support in a great number of ways, and for the guidance and suggestions provided along the completion of this work.

A special mention to Luís Garrote, for always providing me with great advice, fruitful and insightful discussions, by pushing me to go beyond my limits, and for motivating me towards fulfilling my objectives.

I am grateful to the researchers from the projects UID/EEA/00048/2013, AGVPOSYS (CENTRO-01-0247-FEDER-003503) and MATIS (CENTRO-01-0145-FEDER-000014), with FEDER funding, programs PT2020 and CENTRO2020, for all the support they gave me towards the completion of my dissertation, and also to ISR-UC for giving me the conditions to make this dissertation possible.

I am very grateful to all my friends and my girlfriend, for their support and the good moments spent along this journey. Without them I would not have been able to stay the course, and I will always keep with me the memories and experiences lived during this academic journey.

Finally, I am deeply grateful to my family and their continuous support, specially to my mother, that provided me with everything necessary so that I could achieve my masters degree. To everyone mentioned here, and to many others that are not I can only say:

Thank you!

Abstract

Mobile robotics has been constantly researched in the last decades, due to the potential applications it may bring to society. In an industrial context, this subject has been evolving so that mobile robots can be more flexible and precise, without compromising workers' safety. In order for a mobile robot to navigate autonomously, it must be equipped with an environment perception module, so that it can build the environment map, localize itself and plan safe trajectories. Although mapping and localization are two distinct modules, SLAM (Simultaneous Localization and Mapping) techniques apply these two modules in an integrated way.

The main objective of this dissertation was focused on the localization module. The approach intended for this module, included using sensor fusion data applied to a particle filter. The used and studied sensors, included: encoders, a laser scanner, an IMU (Inertial Measurement Unit) and a beacon-based IPS (Indoor Positioning System). These sensors were fused in different ways, so that diverse approaches could be tested.

Experimental tests were conducted in order to assess the performance of the proposed localization approach. These tests included the comparison of the developed localization system with the well known AMCL (Adaptive Monte Carlo Localization) method, analyzing different sensor fusion schemes in different scenarios and in both static and dynamic environments. The results from the experimental tests were analyzed and the main conclusions highlighted the advantages and disadvantages of each sensor fusion method.

Keywords: Sensor Fusion, Localization, Adaptive Monte Carlo Localization, Mobile Robot, Perception.

Resumo

A Robótica Móvel tem sido constantemente investigada nas últimas décadas, devido às potenciais aplicações que pode trazer à sociedade. Em contexto industrial, este assunto tem evoluído, para que os robôs móveis possam ser mais flexíveis e precisos, sem comprometer a segurança dos trabalhadores. De modo a que um robô móvel navegue autonomamente, ele tem que estar equipado com um módulo de perceção do ambiente, para que possa contruir o mapa do ambiente, localizar-se e planear trajetórias seguras. Apesar de mapeamento e localização serem módulos distintos, as técnicas de SLAM (Simultaneous Localization and Mapping), aplicam estes dois módulos de uma maneira integrada.

O objetivo principal desta dissertação foi focado no modulo de localização. A abordagem pretendida para este módulo, incluiu usar dados de fusão sensorial aplicados a um filtro de partículas. Os sensores estudados e usados, incluíram: *encoders*, um IMU (*Inertial Measurement Unit*) e um IPS (*Indoor Positioning System*) baseado em *beacons*. Estes sensores foram fundidos de maneiras diferentes, para que pudessem ser testadas diversas abordagens.

Foram realizados testes experimentais com objectivo de avaliar o desempenho da abordagem de localização proposta. Estes testes incluíram a comparação do sistema de localização com o conhecido método AMCL (*Adaptive Monte Carlo Localization*) e a análise de diferentes esquemas de fusão sensorial em diferentes cenários e em ambos os ambientes, estáticos e dinâmicos. Os resultados dos testes experimentais foram analisados e as principais conclusões evidenciaram as vantagens e desvantagens de cada método de fusão sensorial.

Palavras Chave: Fusão Sensorial, Localização, Adaptive Monte Carlo Localization, Rôbo Móvel, Perceção.

"Creativity is intelligence having fun."

Albert Einstein

Contents

Acknowledgments **i**

Abstract **iii**

Resumo **v**

List of acronyms **xiii**

List of figures **xviii**

List of tables **xix**

1 Introduction **1**

1.1 Context and motivation 2

1.2 Main objectives 4

1.3 Implementations and key contributions 4

2 Localization technologies and methods: State of the art **7**

2.1 Technologies 7

2.2 Localization methods 9

3 Background material **11**

3.1 Particle filter 11

3.1.1 Prediction 12

3.1.2 Update 13

3.1.3 Resampling 15

3.1.4 KLD-sampling 15

3.2 Extended Kalman Filter package 17

3.3 Environment representation 18

3.3.1 Grid-based maps . . . 18

4 Developed work **19**

4.1 MSPF^2 system . . . 19

4.1.1 Initialization section . . . 19

4.1.2 Loop section . . . 20

4.2 KLD-based particle filter algorithm . . . 23

4.2.1 Resampling . . . 25

4.2.2 Prediction . . . 26

4.2.3 Update . . . 29

4.2.4 IPS methods . . . 30

5 Validation platform **33**

5.1 Platform kinematics . . . 34

5.2 Hardware architecture . . . 34

5.2.1 Processing unit . . . 34

5.2.2 Marvelmind system . . . 35

5.2.3 Encoders and RoboteQ motors controller . . . 36

5.2.4 Xsens Mti-G IMU sensor . . . 37

5.2.5 Hokuyo laser scanner . . . 37

5.2.6 Gamepad controller . . . 38

5.3 Sensor data . . . 38

6 Experimental validation **39**

6.1 Methods comparison . . . 41

6.2 Parameters tuning . . . 45

6.3 Large environment validation . . . 46

6.3.1 Scenario 1 . . . 47

6.3.2 Scenario 2 . . . 48

6.4 Recovery tests . . . 50

7 Conclusion and future work **53**

7.1 Conclusion . . . 53

7.2 Future work . . . 54

Bibliography **60**

APPENDICES **62**

A Extra content **62**

List of Acronyms

AGV Automated Guided Vehicle

AMCL Adaptive Monte Carlo Localization

EKF Extended Kalman Filter

GPS Global Positioning System

IC Installation and Maintenance Cost

ID Identification

IMU Inertial Measurement Unit

IPS Indoor Positioning System

IR Infrared

KF Kalman Filter

KLD Kullback–Leibler Distance

MCL Monte Carlo Localization

PF Particle Filter

ROS Robot Operating System

SLAM Simultaneous Localization and Mapping

UKF Unscented Kalman Filter

UWB Ultra-Wideband

List of Figures

1.1 Two different solutions for the AGV navigation system. 1

1.2 Localization systems where (a) represents the setup containing the available "amcl" package in ROS and (b) the expanded system proposed in this dissertation with the modified "amcl" package. 2

1.3 Block diagram of the proposed system where: u^o and u^e are respectively the Odometry and EKF pose data; u^{ip} is the position data from a beacon-based IPS; I^R is the motion data from the IMU sensor; M is a matrix of the map cells data with i and j respectively being the number lines and columns; L is the 2D laser scanner with k being the number of points scanned by the laser (d and α are distance and angle data); $\hat{\mathrm{x}}^R$ is the estimated pose of the PF system. 3

3.1 Particle filter pipeline. 12

3.2 Visual representation of the motion model with x_t^R=[x_t^R y_t^R θ_t^R], x_{t-1}^R=[x_{t-1}^R y_{t-1}^R θ_{t-1}^R] and the dashed line representing the 0º orientation. 12

3.3 Tracing algorithm representation, that encompasses two steps: the algorithm for line tracing and the search for the first occupied cell. In the occupancy-grid map, the gray cells are the ones in the line between state x_t and measurement z_k and the occupied cells are the ones represented in red, with z_k^* representing the first detected occupied cell. 14

3.4 Representation of a particle falling into a grid-map cell, being: (a) Example of a particle falling into an empty bin; (b) Example of a particle falling into an occupied bin. The particle is represented by the black dot (position) and black arrow (orientation). The occupied bins are the red ones while the empty bins are the white ones. The represented histograms have 8 sections/bins. . . 17

3.5 EKF block with inputs and outputs for the used configurations. 18

3.6 Example of an occupancy grid map representation. 18

4.1 MSPF^2 flowchart, including the initialization steps and the cyclic steps that occur in the loop section. 20
4.2 Sensor handler steps, where $\delta_t = \delta_{trans}$ and $\delta_{rot} = \delta_{rot1} + \delta_{rot2}$. 21
4.3 KLD-based particle filter block scheme with: X_t and X_{t-1} being the actual and the previous particle sets; X^*_{t-1} the rearranged previous particle set; $\bar{X}_t$ the actual predicted set; X_0 the initial set; $\hat{\mathrm{x}}^R_t$ the actual estimated pose; M the prior map; z_t the observation; u_t (u^o_t,u^e_t,u^{ip}_t) the controls. Odom is an abbreviation for odometry. 23
4.4 Graphics representing an example of the evolution of the resample stage: (a) Previous particle set, X_{t-1}; (b) Sorted particle set, X^*_{t-1}; (c) Point mass distribution, $f(\mathrm{x}^{*[n]}_{t-1}) = \sum_{j=1}^{n} w^{[j]}_{t-1}$, used to sample a particle for the new particle set. With N being the number os particles in the set and $0 < n \leq 1$. 25
4.5 Verification of available sensor data through a cascade system. 26
4.6 Diagram describing the applied prediction methods based on the available sensor data. 28
4.7 Decision block representing two approaches used in the particle filter to decide between the use of pose or position data. 30
4.8 Example of distance calculation for the verification of one condition for the method. 31

5.1 Platform is composed by: (1) Processing Unit; (2) Marvelmind Beacon; (3) Gamepad controller; (4) IMU sensor; (5) Lead-Acid Batteries; (6) Wheels with encoders; (7) Laser Scanner; (8) RoboteQ Motors Controller. 33
5.2 Representation of the AGVs motion model. 34
5.3 Hardware components of the Marvelmind system with: (a) Modem that communicates with all beacons and the processing unit; (b) Beacons used for trilateration with one staying in the AGV and the rest being scattered throughout the environment. 35
5.4 ISR-UC floor 0 beacons map configured on the Marvelmind software. 36

6.1 ROS schematic for the MSPF^2 localization method, which contains the ROS nodes used in the system, the subscribed and published topics, and the hardware that communicates with mentioned nodes. 40

6.2 ROS schematic for the modified AMCL method, which contains the ROS nodes used in the system, the subscribed and published topics, and the hardware that communicates with mentioned nodes. 40
6.3 Picture of the ISR-UC experiments room. 41
6.4 Occupancy grid map representing the ISR-UC experiments room with the defined origin of the axis marked as the point (0,0). The intended trajectory is marked by the red arrows. 41
6.5 Representation of the sensors' and EKF data overlapped with the intended testing trajectory. 42
6.6 Overlap of the trajectories given by the MSPF^2 and AMCL methods with the intended path. 43
6.7 Occupancy grid map representing the floor 0 of the DEEC-UC with the defined origin of the axis marked as the point (0,0). The pictures are referent to the arrows with the same color as its contour (*e.g.*, red arrow refers to the picture with the red contour). 46
6.8 Representation of the overlapped trajectories from 10 samples of all the MSPF^2 methods. Area 1 is a zoom of a curve while Area 2 is a zoom of the initial and final position. 47
6.9 Part of the grid cell map used for this test, where A and B are respectively the initial and the final point of the corridor. 48
6.10 Representation of the overlapped trajectories from ten samples of all the MSPF^2 methods, with the red points being the initial points of each trajectory and the red ones, the last points. 49
6.11 Recovery test performed with the "MS(Odom)" method. The snapshots are taken from the ROS rviz. 51
6.12 Recovery test performed with the "MS(Odom+IPS1)" method. The snapshots are taken from the ROS rviz. 51
6.13 Recovery test performed with the "MS(Odom+IPS2)" method. The snapshots are taken from the ROS rviz. 52

A.1 Blueprint of the ISR-UC floor 0. 62
A.2 Score's RMS of the tests performed for different values of the α_1, α_2, α_3 and α_4 parameters. 63

A.3 Score's RMS of the tests performed for different values of the Δd and $\Delta\theta$ parameters. 63
A.4 Score's RMS of the tests performed for different values of the α^{ip} parameter. 64
A.5 Score's RMS of the tests performed for different values of the th^{ip} parameter. 64
A.6 Score's RMS of the tests performed for different values of the maximum number of particles parameter. 65

List of Tables

2.1 Technical specifications of some indoor localization technologies (adapted from [18, 22]). IC: installation and maintenance cost. 9

2.2 Brief review of systems with indoor localization methods. 10

4.1 Table containing the operation mode possibilities and the respective available sensor information. 27

5.1 Technical specifications of the Marvelmind system adapted from [51] 36

5.2 Hokuyo's UTM-30LX laser main specifications [49]. 38

5.3 Sensors computational times. 38

6.1 Table containing the RMS and the mean of the two different metric evaluations made on tested methods. The values in bold are the three best values of each column. 44

6.2 Data from the tuning tests, including the evaluation values made with Metric 1, the experimented ranges and the Best Value for for each parameter. The proposed testing range was between the Min and Max values defined on the table. 45

6.3 Table containing the evaluation data for the trajectories given by the different MSPF^2 methods. This data is relative to the mean of the points evaluation (with M1) of each trajectory. The numbers in bold represent the three best values in each column. 48

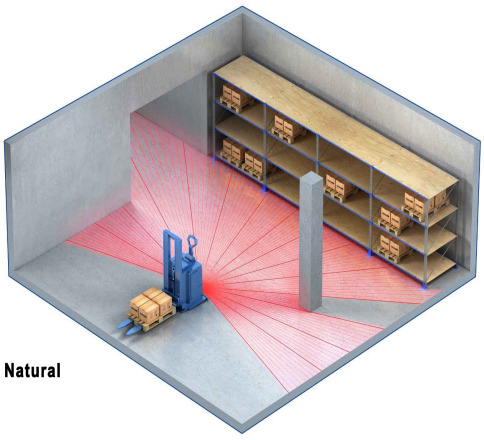
6.4 Table containing the evaluation data for the trajectories given the different MSPF^2 methods on the corridor. This data is relative to the mean of the points evaluation (with M1) of each trajectory. The numbers in bold represent the three best values in each column. 50

Chapter 1

Introduction

Robotic systems have been developed to execute all kinds of industrial tasks (*e.g.*, production) [1], with the purpose of providing optimization and flexibility to different processes. A robotic vehicle that gained some importance in the industrial field is the Automated Guided Vehicle (AGV), since its purpose is the transportation of materials in industrial environments (*e.g.*, warehouses) [2]. There are many solutions based on different sensors for localization or path planning of AGVs, where some of the most deployed solutions are magnetic tape tracking and laser-based localization [2].

(a) Magnetic tape tracking.[1]



(b) Laser-based localization.[2]

Figure 1.1: Two different solutions for the AGV navigation system.

[1] https://automotivemanufacturingsolutions.com/wp-content/uploads/2016/04/AGV-seat-1.jpg

[2] https://www.transbotics.com/hubfs/Automatic_Guided_Vehicle_natural_navigation.jpg?t=1502905426963

1.1 Context and motivation

One of the potential approaches for industrial AGVs is laser-based localization methods but most of the actual localization methods rely on multimodal sensor data, because laser-based systems alone, have problems in accuratly localizing the AGV in featureless zones (*e.g.*, corridors) [3, 4], or in zones with dynamic obstacles. Among the used sensors for localization in industrial settings, some of the most popular are the laser scanner, wheel encoders (used for odometry) and the Inertial Measurement Unit (IMU).

Within laser-based approaches, the particle filter (PF) based localization approach have a low software implementation overhead [5], but sometimes, due to their lack of sensor information, the particles start to diverge [6]. One of the most popular among these methods is the AMCL (Adaptive Monte Carlo Localization) [7], which uses pose (position and orientation) data (*e.g.*, odometry), a prior cell grid map and laserscanner data to estimate the robot's pose (Fig. 1.2 (a)).

Open versions of the aforementioned algorithms are available in Robot Operating System (ROS) [8]. Since the available "amcl"[3] software package could only receive odometry data (excluding laser and map data), one of the motivations was to modify the package, so it can support another similar message (pose type message). This message can be generated by combining and filtering data from two sensors (encoders and IMU), or simply filtering data of the already used sensor in an Extended Kalman Filter (EKF), as shown in Fig. 1.2 (b). Here, the used EKF software is also a software package in ROS, named as "robot_localization"[4].

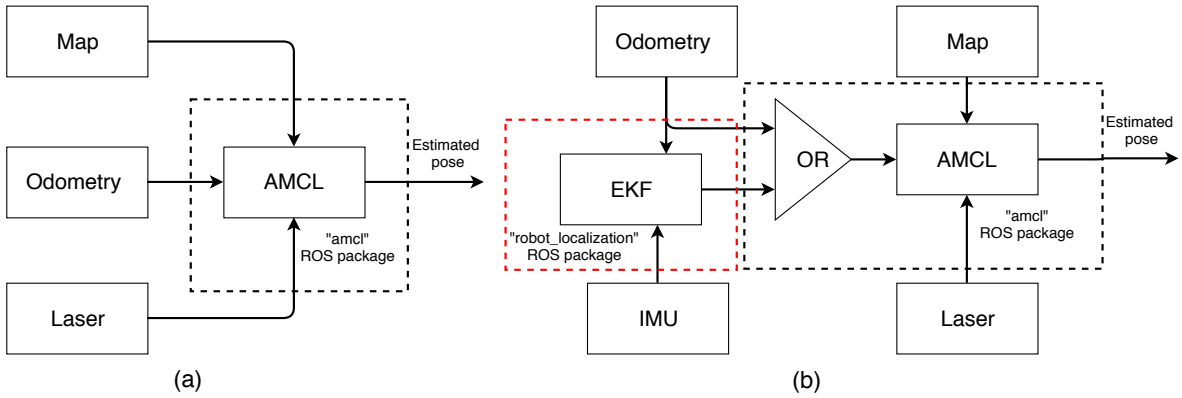
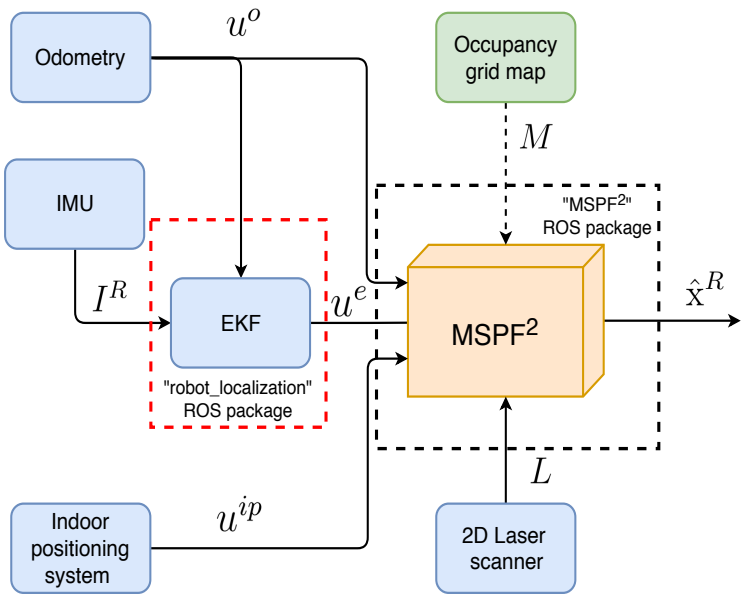


Figure 1.2: Localization systems where (a) represents the setup containing the available "amcl" package in ROS and (b) the expanded system proposed in this dissertation with the modified "amcl" package.

[3] http://wiki.ros.org/amcl

[4] http://wiki.ros.org/robot_localization

Although these modifications (Fig. 1.2 (b)) help on the study of the localization system, they are not enough to solve some of the problems faced by mobile robots, such as: dynamic obstacles, featureless zones and localization failures. So, the main motivation of this dissertation was to develop a system using multi-modal sensor data (as shown in Fig. 1.3) that uses position and pose (position and orientation) data from multiple sensors and study which would be the approach to be applied in a PF-based multi sensor fusion algorithm, to make the localization more robust and reliable.



Variables	Decomposition
$\hat{x}^R$	$[\hat{x}^R\ \hat{y}^R\ \hat{\theta}^R]^T$
u^o	$[x^o\ y^o\ \theta^o]^T$
u^e	$[x^e\ y^e\ \theta^e]^T$
u^{ip}	$[x^{ip}\ y^{ip}]^T$
I^R	$[\ddot{x}^R\ \dot{\theta}^R]^T$
L	$\begin{bmatrix} d_1 & d_2 & \cdots & d_k \\ \alpha_1 & \alpha_2 & \cdots & \alpha_k \end{bmatrix}$
M	$\begin{bmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,j} \\ m_{2,1} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ m_{i,1} & \cdots & \cdots & m_{i,j} \end{bmatrix}$

Figure 1.3: Block diagram of the proposed system where: u^o and u^e are respectively the Odometry and EKF pose data; u^{ip} is the position data from a beacon-based IPS; I^R is the motion data from the IMU sensor; M is a matrix of the map cells data with i and j respectively being the number lines and columns; L is the 2D laser scanner with k being the number of points scanned by the laser (d and α are distance and angle data); $\hat{x}^R$ is the estimated pose of the PF system.

The proposed PF block (shown in Fig. 1.3) is named as Multi Sensor Particle Filter Fusion (MSPF^2), is the main result of this dissertation work and it is thoroughly described in Chapter 4. The odometry, 2D laser scan, Marvelmind beacon-based Indoor Positioning System (IPS) and IMU are briefly described in Section 5.2. An overview of the "robot_localization" ROS package and the occupancy grid map are is given in Chapter 3.

1.2 Main objectives

As mentioned in the previous section, the main objective of this dissertation work was the development of a localization approach that could integrate multi-modal sensor data in a PF based method for mobile robot's pose estimation. More precisely the main objectives can be divided in two steps as follows:

- **Multi-modal sensor system:** The first step is to develop a system that can receive data from multiple sensors, process this data according to its type (pose or position) and send it to the PF;
- **PF based sensor fusion:** The filter implementation needs to guarantee that data from multiple sensors can be fused, in order to be used for robot pose estimation. Different approaches of sensor fusion need to be tested in order to find a method that is robust, precise and that can estimate the robot's pose in both static and dynamic environments.

1.3 Implementations and key contributions

The main implementations and contributions are described in:

Developed Work (Chapter 4):

- **Multi sensor fusion system design:** Description of the developed software design for multiple sensor inputs, including the local and global localization sensors.
- **Particle filter:** Detailed description of the MSPF^2 algorithm, including the tested sensor fusion methods.

Validation Platform (Chapter 5):

- **Hardware architecture:** Overview of the different onboard sensors for the validation of the MSPF^2 methods.
- **Teleoperation:** Two methods were deployed in ROS to directly steer the AGV platform with a game controller. One of the methods consists on directly sending the commands from the gamepad controller to the processing unit, while the other sends

the commands to a PC, which forwards them to the platforms processing unit trough the same network.

- **Global positioning system:** Beacons, for indoor global positioning (the IPS system), were inserted in the robot and environment, so that global positioning data could be acquired.

Experimental Validation (Chapter 6):

- **Parameters tuning:** An experimental tuning was performed on the PF parameters and tested on the same conditions, to ensure the selection of the best parameters for the mobile robot.

- **Off line testing:** Different environments were tested in different situations with the methods that showed best performance, using the data from several datasets.

Chapter 2

Localization technologies and methods: State of the art

Indoor localization is a very important research topic in an industrial context because it is important for mobile robots to be developed with trajectory flexibility and reduction of costs (robot and associated setup cost) in mind [3, 9, 10]. The majority of the high-level tasks performed by a robot (*e.g.*, planing, execution) are based on the supposition that the robot can answer to three basic questions: "Where am I?", "Where am I going", "How can I get there?". The first question is related to the localization problem and the latter two, to the knowledge of the robot's workspace and decision-making [11]. In an industrial context, an AGV needs a precise pose, so that it can execute its tasks with the maximum possible safety in order to complete its mission and guarantee that the other agents (workers and other AGVs) can also execute their tasks safely [2].

2.1 Technologies

Some of the most used technologies for AGV localization in indoor environments are: IMUs, encoders, IPSs and laser scanners [12, 13, 14]. There is also the magnetic tape tracking system, which does not locate the AGVs but allows them to follow intended paths [15], and optical line guidance sensors[1], that are used for the same purpose. In AGV or mobile robots localization, the IPS presents important features since this technology is related to absolute/global positioning. The IPS is based in principles that somehow resemble a Global Positioning System (GPS), where in a simplified way a constellation of satellites (beacons

[1] `https://www.sick.com/us/en/line-guidance-sensors/optical-line-guidance-sensors/c/g466252`

with known positions) send distance measurements and one or more mobile devices receive these measurements so that they can obtain the robot's position by means of some known techniques, such as trilateration [16, 17]. The use of IPS requires changes to the work environment, *i.e.*, beacons must be placed in the surrounding structures in key positions so as to avoid reflection or occlusion problems and to increase the accuracy of measured distances.

For indoor localization, several systems are based in technologies such as ultra-wideband (UWB), infrared (IR), Wi-Fi and ultrasound [18].

Systems based in UWB use pulse transmissions short in time over a higher bandwidth of frequencies (>500MHz) instead of a specific frequency to transmit information thus achieving a lower consumption of the components. Applications with these systems consist in a mobile element detecting the emitted pulses by the beacons placed in the infrastructure [19, 20].

With IR based systems, IR emitters are installed in known positions on the robot's workspace where each transmission is codified by a frame with a specific identification (ID), the beacon position and other elements. One of the greatest problems of these systems is the necessity for emitters and receivers to be in the same field of vision. This is due to the restrict angle of IR transmission, which for wide areas, can demand an higher number of beacons to ensure a robust positioning. Another major problem of this technology is the interference in the IR signal [21].

Systems based in existent Wi-Fi networks can be used for position estimation of any agent with a good precision [22]. This type of localization can easily be applied in several indoor scenarios. However, problems related with the network congestion and interferences caused by elements in the scenario, can limit the accuracy of this type of systems [20, 23]

In ultrasound-based systems, sound waves transmission are effectuated on the non-audible specter (superior to 15KHz). The localization is performed based on the time of flight between a receptor and an emitter, being necessary the obtaining of at least two readings coming from different emitters to be able to calculate a robust position. Depending on the scenarios and on the types of materials used on the walls, one of the greatest problems of this kind of systems is the blocking and reflection of ultrasound signals. Additionally, it is important to note that the emitter and receptor are in line of sight to get a more accurate position [24, 25].

Table 2.1 presents some technical specifications of the technologies described in this chapter, in particular, accuracy and coverage values.

Technologies	UWB	IR	Wi-Fi	Ultrasound
Accuracy	15cm	57cm – 2.3m	1.5m	1cm – 2m
Coverage (m)	1-50	1-5	20-50	2-10
IC	High	High	Low	High

Table 2.1: Technical specifications of some indoor localization technologies (adapted from [18, 22]). IC: installation and maintenance cost.

2.2 Localization methods

Within localization methods for mobile robots some of the well-known methods are based on the following stochastic filters frameworks: the EKF, the Unscented Kalman Filter (UKF), the Monte Carlo Localization (MCL) and the adaptive PFs.

Mobile robots can not always be described by linear state transitions but the KF (Kalman Filter) assumes this linearity, and for this reason, is only applicable to the simplest robotic problems [7]. The linearity assumption is overcome by the EKF, *i.e.*, it uses the *Taylor expansion* (first order) to find a linear approximation of nonlinear functions [7].

The EKF is a filtering strategy that was used over the last decades but it is consensual that it presents some difficulties (*e.g.* difficult implementation and tuning) in tracking and control fields. It also assumes that the distributions of noise sources are Gaussian, which is not assumed by the UKF [26]. The UKF uses the unscented transformation for statistics calculation of random variables that experience a nonlinear transformation [27]. This filter can be used for mobile robot's applications such as multi-sensor-based human detection and tracking system [28].

In the MCL PF method, the representation of the probability density function is made by maintaining a set of samples (particles) that are randomly drawn from this density function [29]. This method is able to represent multi-modal distributions and it can be applied to both local and global localization problems [7]. PFs are easily implemented, which makes them an attractive solution for localization methods of mobile robots. This method also has some limitations, such as, the impossibility of solving the kidnapped robot problem[2] in the method's original form since there might be no samples around the robot's new pose after it has been transported [4].

[2]The kidnapped robot problem consists on locating the robot after it has been transported to another location of the environment without any knowledge of this transportation.

In adaptive PF approaches such as the AMCL, the sample set size is determined in each iteration, based on statistical bounds. The particle's set size varies depending on the faced problem (*e.g*, when facing position tracking situations, the particle filter uses a small sample set, but when facing global localization problems, it uses a larger sample set [30]). This approach has some associated improvements over PFs with sample sets of constant size, such as improvements on the computational overhead, position tracking and global localization problems [30, 31].

Table 2.2 contains a brief review of some localization methods deployed in indoor mobile robots and other systems over the years.

Table 2.2: Brief review of systems with indoor localization methods.

Localization methods	**Description**	**Technologies**
Extended Kalman Filter (EKF) based localization applied to Robox (2002) [32].	Social guide robot at Expo 02, guided 686.000 people over 5 months, seven days a week, up to twelve hours a day. Enhanced navigation system, that handles the exhibition environment, which was highly dynamic because of the people walking around.	Encoders Two laserscanners
Localization method based on the MCL (Monte Carlo Localization) approach used for in robot Jinny (2004) [33].	The Jinny has been tested in various environments like an office building in KIST (Korea Institute of Science and Technology). The main functions of this robot consist on human interaction and autonomous navigation.	Encoders Two laserscanners Gyroscope
The robot PR2 (2010) [34] dynamically switched between the AMCL and the EKF localization methods depending on the available data.	Indoor robot for social environments, which required a navigation system that was capable of operating in an office without having any incidents for 26.2 miles. Performed in a marathon, without human interaction for over 30 hours.	Encoders Laserscanner IMU
Use of IMU and UWB sensor fusion applied to a particle filter for postion estimation of mobile agents (2017) [35].	System composed by modules placed on the users torso, which gives relative positioning between multiple mobile users. This system has many applications, as it is in the case of search and rescue disaster areas, or social interaction scenarios.	IMU UWB

Chapter 3

Background material

3.1 Particle filter

"The particle filter is an alternative nonparametric implementation of the Bayes filter", a concept extracted from [7]. This algorithm begins by randomly distributing a set of samples across the robot's workspace, then, in each iteration there is an attempt to approximate the particles to the real system state. The particles of this filter are denoted as:

$$X_t = [\mathrm{x}_t^{[1]}, \mathrm{x}_t^{[2]}, ..., \mathrm{x}_t^{[N]}] \tag{3.1}$$

Every $\mathrm{x}_t^{[n]}$ (with $1 \leq n \leq N$) particle represents a hypothesis of the real state of a system at time t, with their respective importance factor ($w_t^{[n]}$). N represents the number of samples in the set X_t, which is a fixed number of states, but in some methods it can be a variable, as it happens in the KLD-sampling method [30], that is described in this dissertation.

PFs (Fig. 3.1) use the posterior state of the particles (X_{t-1}), the control (u_t) and measurement (z_t) at time t to estimate the new particle states. PFs can be decomposed in 3 fundamental steps: Prediction; Update; Resampling.

In the literature, the authors of [29, 35, 36, 37] and others, mention these steps with different names. The name chosen for the first step is "Prediction", because in it the next state of the particles is predicted based on the motion model of the mobile robot. The second step updates the weight of the particles based on the observation model of the sensors, therefore it is named "Update".

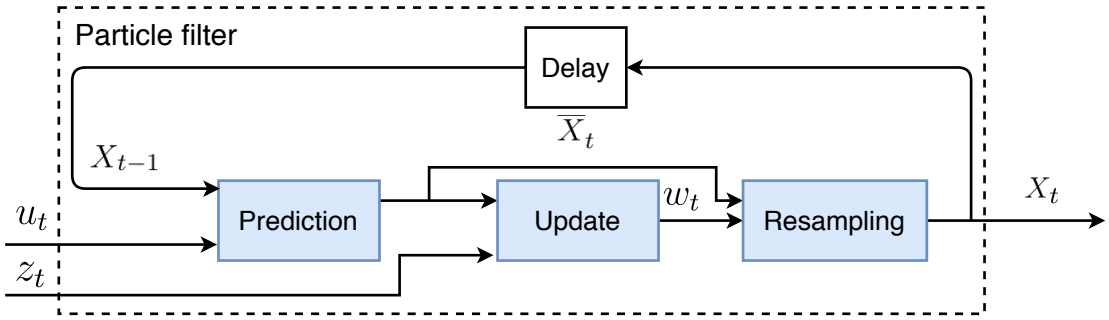


Figure 3.1: Particle filter pipeline.

Before entering in the iterative steps of the PF, there must be an initialization of the parameters. The initial belief $bel(\mathrm{x}_0)$ can be obtained by generating N random particles based on the prior distribution $p(\mathrm{x}_0)$ and attributing to each particle the same importance factor $\frac{1}{N}$. One of the objectives of this dissertation is to use the PF in a differential mobile robot [38]. For this reason, the descriptions about the filter will no longer be generic, but will be directed to the mobile robot used in the development of this dissertation.

3.1.1 Prediction

In the prediction stage, the PF generates an hypothetical pose $\mathrm{x}_t^{[n]}$ of the platform based on the prior particle $\mathrm{x}_{t-1}^{[n]}$ and the control u_t. The particles are decomposed in 3 states $(x^{[n]}, y^{[n]}, \theta^{[n]})$ and the control u_t is a sensor estimation of the mobile robot's pose at time t ($u_t = [\bar{x}_t^R, \bar{y}_t^R, \bar{\theta}_t^R]$). As it is suggested in [7] the previous and actual pose estimations are transformed in a translation (δ_{trans}) and 2 rotations (δ_{rot1} and δ_{rot2}) as it is shown in Fig. 3.2.

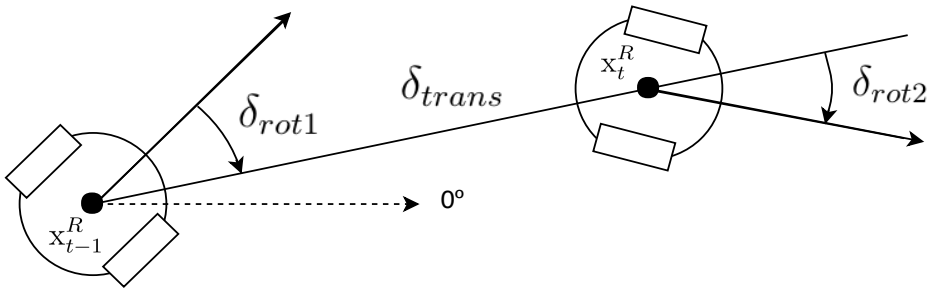


Figure 3.2: Visual representation of the motion model with x_t^R=[x_t^R y_t^R θ_t^R], x_{t-1}^R=[x_{t-1}^R y_{t-1}^R θ_{t-1}^R] and the dashed line representing the 0º orientation.

The following equations are applied in order to obtain the prediction of all the hypothetical poses (particles) [7]:

$$\begin{cases} \delta_{rot1} = atan2(\bar{y}_t{}^R - \bar{y}^R_{t-1}, \bar{x}_t{}^R - \bar{x}^R_{t-1}) - \bar{\theta}^R_{t-1} \\ \delta_{trans} = \sqrt{(\bar{x}_t{}^R - \bar{x}^R_{t-1})^2, (\bar{y}_t{}^R - \bar{y}^R_{t-1})^2} \\ \delta_{rot2} = \bar{\theta}^R_t - \bar{\theta}^R_{t-1} - \delta_{rot1} \end{cases} \tag{3.2}$$

$$\begin{cases} \hat{\delta}_{rot1} = \delta_{rot1} - \mathbf{sample}(\alpha_1\delta^2_{rot1} + \alpha_2\delta^2_{trans}) \\ \hat{\delta}_{trans} = \delta_{trans} - \mathbf{sample}(\alpha_3\delta^2_{trans} + \alpha_4\delta^2_{rot1} + \alpha_4\delta^2_{rot2}) \\ \hat{\delta}_{rot2} = \delta_{rot2} - \mathbf{sample}(\alpha_1\delta^2_{rot1} + \alpha_2\delta^2_{trans}) \end{cases} \tag{3.3}$$

$$\begin{cases} x^{[n]}_t = x^{[n]}_{t-1} + \hat{\delta}_{trans}\cos(\theta^{[n]}_{t-1} + \hat{\delta}_{rot1}) \\ y^{[n]}_t = y^{[n]}_{t-1} + \hat{\delta}_{trans}\sin(\theta^{[n]}_{t-1} + \hat{\delta}_{rot1}) \\ \theta^{[n]}_t = \theta^{[n]}_{t-1} + \hat{\delta}_{rot1} + \hat{\delta}_{rot2} \end{cases} \tag{3.4}$$

The relative motion parameters $(\delta_{rot1}, \delta_{trans}, \delta_{rot2})^T$ are obtained by applying (3.2). Then, the relative motion parameters $(\hat{\delta}_{rot1}, \hat{\delta}_{trans}, \hat{\delta}_{rot2})^T$ are obtained with (3.3) by subtracting sampled error to each motion parameter, with the error parameters being α_1, α_2, α_3 and α_4. The new states $(x^{[n]}_t, y^{[n]}_t, \theta^{[n]}_t)$ are computed with (3.4) by adding the relative motion parameters to the previous states. One approach for error sampling is the Box-Müller method [39, 40]:

$$\mathbf{sample}(\sigma_e) = \sigma_e \cdot r_2\sqrt{-2\ln(r_1)} \tag{3.5}$$

with $r_1 \sim \mathcal{U}(-1, 1)$, $r_2 \sim \mathcal{U}(-1, 1)$ and σ_e being the input of the function. $\mathcal{U}(-1, 1)$ is an uniform distribution.

3.1.2 Update

There is a great variety of different sensor modalities for robots, such as tactile sensors, range sensors, or cameras. Within the ones used in mobile robotics, range finders, which are some of the most popular, provide measurements of distance to nearby objects. The range may be measured along a beam, which is a good model for laser range finders, or within a cone, which is the preferable model of ultrasonic sensors. The measurement model is formally defined as a conditional probability distribution $p(z_t|\mathrm{x}_t, M)$, being x_t a particle, z_t the measurements at time t, and M the map of the environment, with this being usually a grid cell map. The following algorithm refers to a measurement model for beam range finders that can be used in the update step.

Algorithm 1: Range beam model algorithm adapted from [7].

Data: beam range measurements $z_t = \{z_1, ..., z_K\}$, particle $\mathrm{x}_t = \{x_t, y_t, \theta_t\}$, occupancy grid map M

```
w = 0;
for k=1 to K do
    z_k^* = tracing_algorithm(x_t, z_k, M);
    w = w + N_z(0, z_k^* - z_k);
end
w = w/K; return w
```

In Algorithm 1, for each distance measured by the range finder, the algorithm searches for a traced distance (distance to the first occupied cell), by searching the grid map cells in the line between the robot and the measured point. Based on the difference between the traced and measured distances, the weight of the particle being processed is calculated. One approach for tracing the cells in the line between the state x_t and measure z_k, is the Bresenham line algorithm described in [41]. This method computes the cells in a line between the robot's pose and the measured point. After using the Bresenham line algorithm, the cells are analyzed in order to find the first occupied cell in that line. This process is shown in Fig. 3.3, where x_t is a possible state (particle), z_k is the measured point by the sensor and z_k^* is the point of the first occupied cell of the line.

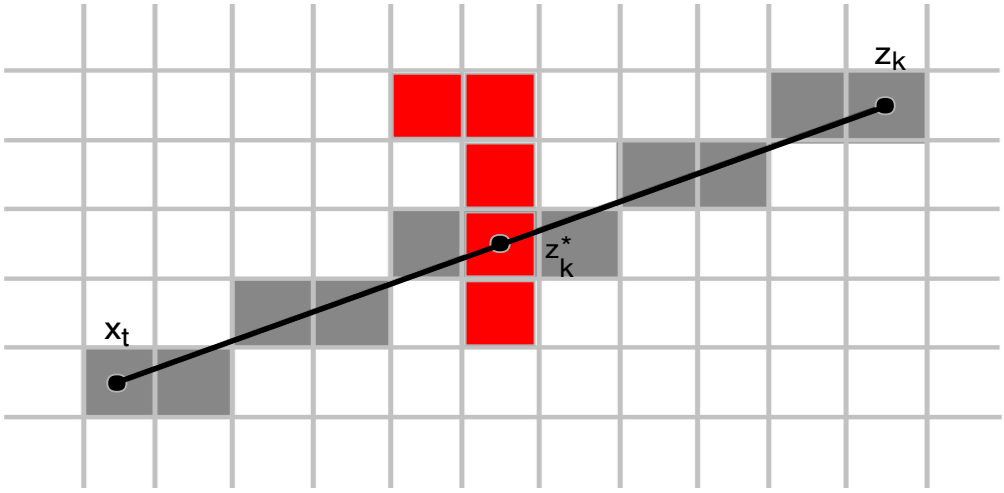


Figure 3.3: Tracing algorithm representation, that encompasses two steps: the algorithm for line tracing and the search for the first occupied cell. In the occupancy-grid map, the gray cells are the ones in the line between state x_t and measurement z_k and the occupied cells are the ones represented in red, with z_k^* representing the first detected occupied cell.

3.1.3 Resampling

In the resampling step, the temporary particle set $\bar{X}_t$ of N particles is transformed into another set (X_t) of identical size. The algorithm draws particles from the temporary set $\bar{X}_t$, with the probability of drawing a particle being related to the particle weight. Depending on the resample method, the resampled set of particles can contain duplicates, which in some cases happens because particles from the previous distribution are randomly sampled and this sampling can chose the same particle more than once [42, 43, 44]. One method for the resampling of particles, is the multinomial resampling [44]. This method starts by ordering the particles according to their weights. Then, N numbers r are generated as follows:

$$r^{[n]} \sim \mathcal{U}[0,1],\ with\ n = [1,2,...,N] \tag{3.6}$$

these numbers are used to select a particle $\mathrm{x}_t^{[n]}$ according to the multinomial distribution as follows:

$$\mathrm{x}_t^{[n]} = \bar{\mathrm{x}}_t^{[j]},\ with\ j = F^{-1}(r^{[n]})\ and\ r^{[n]} \in [\sum_{s=1}^{j-1} w_t^{[s]}, \sum_{s=1}^{j} w_t^{[s]}] \tag{3.7}$$

where $\bar{\mathrm{x}}_t^{[j]}$ is the particle from the temporary set $\bar{X}_t$ with index j obtained from the inverse cumulative probability distribution (F^{-1}) of the normalized particle weights.

3.1.4 KLD-sampling

The Kullback–Leibler Distance (KLD) sampling is a method for adapting the particle set size of particle filter algorithms. In this method, instead of applying each stage separately to the entire particle set, each particle goes trough all stages so that the number of desired samples (n_χ) can be updated each time a particle is processed. When the number of desired samples/particles is reached, the particles are not sampled anymore until the next filter iteration. To update the number of desired samples, the number k of occupied bins, needs to be known. For this purpose, each cell of the occupancy-grid map will contain an angle histogram divided into n sections and each of these n sections is considered a bin (b). As Algorithm 2 shows, the difference from this to the other methods is in lines 9 to 13 (KLD condition). These lines contain the verification of the bin where the particle fell, the update of the number of occupied bins (k) and the step that updates the number of desired particles based on the number of occupied bins. In line 12 of Algorithm 2, the number of desired samples is computed trough the presented equation, where $z_{1-\delta}$ is the upper $1-\delta$ quantile of the standard normal distribution [30]. Figure 3.4 illustrates how the bins are verified.

Algorithm 2: KLD-sampling algorithm adapted from [30].

Data: Posterior particle set $X_{t-1} = \{(\mathrm{x}_{t-1}^{[n]}, w_{t-1}^{[n]}) |\ n = 1, ..., N\}$, representing belief $bel(\mathrm{x}_{t-1})$, control u_t, observation z_t, bounds ε and δ, minimum number of samples n_{min}

$X_t = \emptyset$, $n = 1$, $n_\chi = 0$, $k = 0$, $\alpha = 0$; ; // Initialize

$X_{t-1} = \mathrm{Resample}(X_{t-1})$; ; // Rearranges the previous particle set

do

  $\mathrm{x}_{t-1}^{[j]} = \mathrm{Draw}(X_{t-1})$; // Samples a particle with index j from X_{t-1}

  $\mathrm{x}_t^{[n]} = p(\mathrm{x}_t^{[n]} | \mathrm{x}_{t-1}^{[j]}, u_{t-1})$; // Predicts next state

  $w_t^{[n]} = p(z_t \mid \mathrm{x}_t^{[n]})$; // Computes Importance weight

  $\alpha = \alpha + w_t^{[n]}$; // Updates normalization factor

  $X_t = X_t \cup \{\mathrm{x}_t^{[n]}, w_t^{[n]}\}$; // Inserts new particle in the new particle set

  if $\mathrm{x}_t^{[n]}$ *is in empty bin* b **then**

    $k = k + 1$; // Updates number of non-empty bins

    b = occupied; // Marks bin as non-empty

    $n_\chi = \frac{k-1}{2\varepsilon}(1 - \frac{2}{9(k-1)} + \sqrt{\frac{2}{9(k-1)}} z_{1-\delta})^3$; // Updates bin size

  end

  $n = n + 1$;

while *(*$n < n_\chi$ *and* $n < n_{min}$*)*;

for $i = 1, ..., n$ **do**

  $w_t^{[i]} = w_t^{[i]} / \alpha$; // Normalizes the weight of each particle

end

return X_t

In Fig. 3.4 (a) the sampled particle falls into an empty bin, which turns into an occupied bin and in this situation, the number k is updated. In Fig. 3.4 (b) the bin is already occupied, so nothing happens. The bin in which the particle falls into is determined by the orientation of the particle as shown in Fig. 3.4. Since the KLD-sampling is a method for adaptive particle filter algorithms, it brings some advantages, such as improvements on the computational overhead, as explained in Section 2.2.

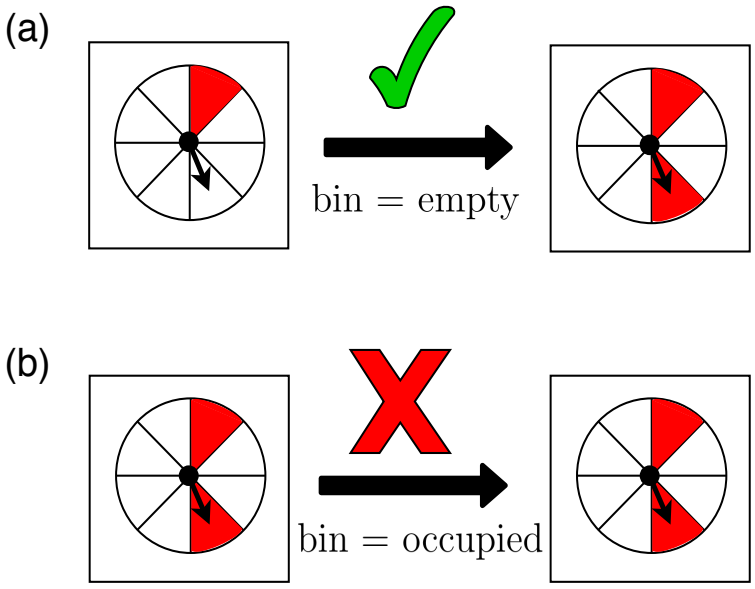


Figure 3.4: Representation of a particle falling into a grid-map cell, being: (a) Example of a particle falling into an empty bin; (b) Example of a particle falling into an occupied bin. The particle is represented by the black dot (position) and black arrow (orientation). The occupied bins are the red ones while the empty bins are the white ones. The represented histograms have 8 sections/bins.

3.2 Extended Kalman Filter package

In order to test different methods of sensor fusion, multiple sources of pose or position data were needed. The "robot_localization" ROS package[45] presented some advantages towards the proposed system. This software can fuse data from multiple sensors in an EKF, or simply apply information from one sensor to the filter. The configurations used for this system permitted to filter the IMU data fused with odometry data, or just the odometry in the EKF. Figure 3.5 represents the inputs and outputs block of the ROS package. The EKF was chosen to be used instead of the KF (Kalman Filter) because it determines the next state and measurement probabilities trough nonlinear transitions [7].

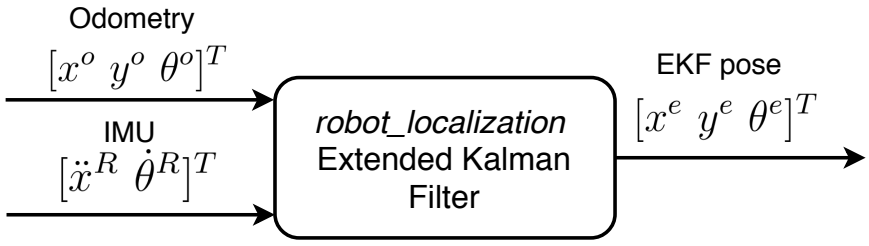


Figure 3.5: EKF block with inputs and outputs for the used configurations.

The KF assumes linear state transitions and measurements, but a mobile robot can not always be described by linear state and/or measurement transitions.

3.3 Environment representation

In order for a mobile robot to be autonomous, it needs to be able to use a map representing the environment. This map could be either built using different types of sensor data, or it could be priorly given. There are 2 types of representations addressed in [46], the topological maps and the grid-based (metric) maps, but in this dissertation only grid-based maps were used.

3.3.1 Grid-based maps

Occupancy-grid maps can represent the environment in 2D or 3D grids, but in this dissertation, only 2D representations were used. In this kind of representation, the environment is divided into a matrix of cells that form the metric grid, where the resolution depends on the cell size. The cells have associated an occupancy value that represents the probability of the cell being occupied. Figure 3.6 shows an example of a grid-based map.

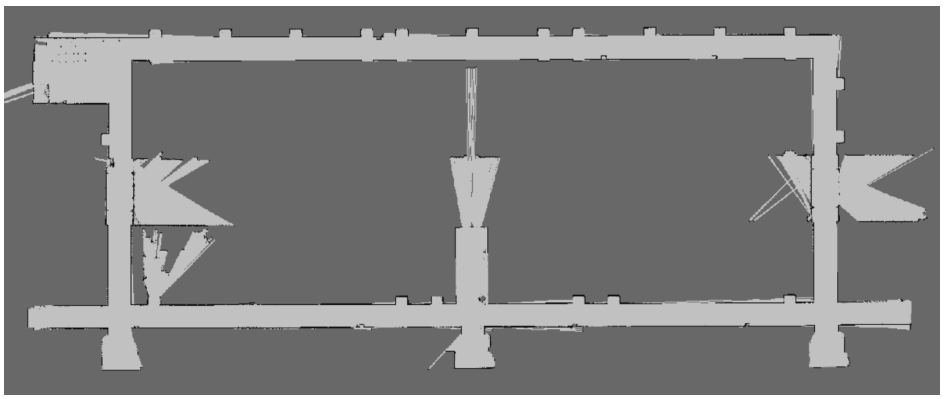


Figure 3.6: Example of an occupancy grid map representation.

Chapter 4

Developed work

This chapter presents the proposed MSPF^2 localization system, with more emphasis on the system design and the PF algorithm.

4.1 MSPF^2 system

The PF used in this dissertation was inspired in the methods described in [7] and [30]. However, the developed system differs from the existing ones in the sense that it is able to receive pose and/or position information from multiple sensors (*i.e.*, laser scanner , encoders and IMU).

4.1.1 Initialization section

The MSPF^2 approach can be decomposed in two main sections, as shown in Fig. 4.1, the initialization and the loop. The initialization consists on two procedures, as follows:

- **Map loading**: There is a need for a map of the environment to validate the observations coming from the laser scan, so the first step is to load the occupancy grid map file. This file is generated from a version of the HectorSLAM[1][47], and it is coded in yaml format.

- **Parameters configuration**: To avoid wasting resources in redundant localization estimations, a configuration is made for each sensor that provides pose data, which only lets the localization method work when a defined minimal translation or rotation is attained by sensor measurements. The function that governs this step receives as

[1] `http://wiki.ros.org/hector_slam`

inputs: a key string, a value of minimum distance (Δd), and a value of minimum rotation ($\Delta\theta$). The key string serves to create variables for each sensor, so that their data is processed independently, while Δd and $\Delta\theta$ values serve to preclude the sensor data from being used unless one of those values of displacement have been reached since the last estimated pose.

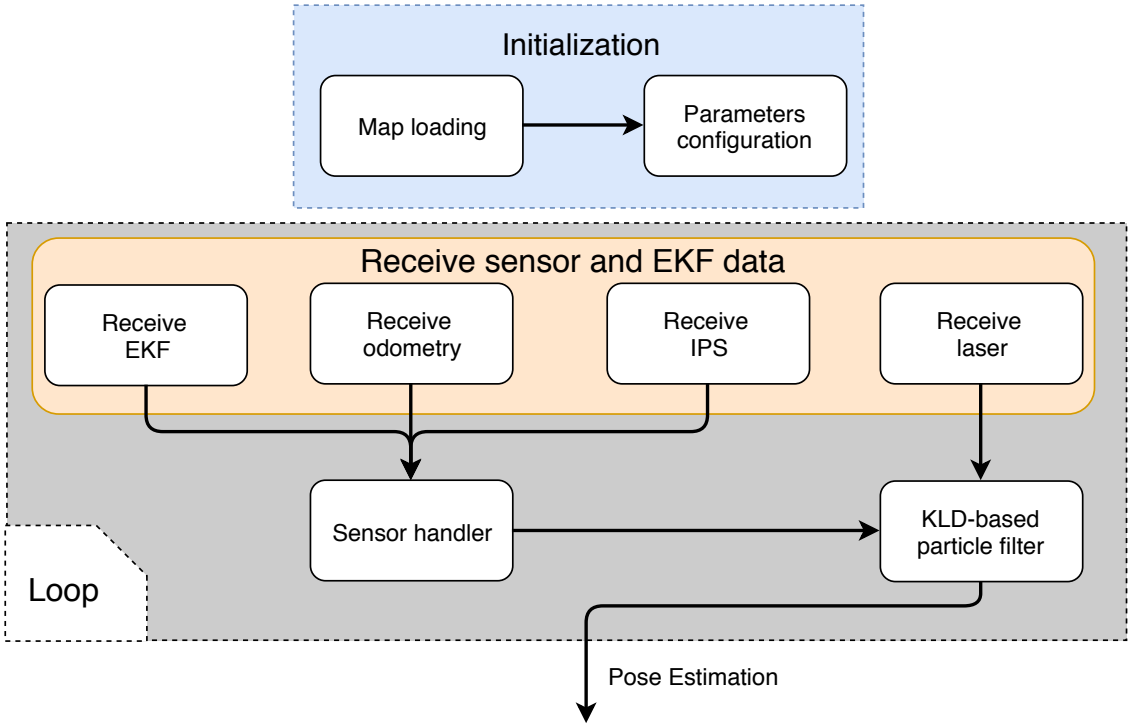


Figure 4.1: MSPF^2 flowchart, including the initialization steps and the cyclic steps that occur in the loop section.

4.1.2 Loop section

The loop section is where the sensor information is gathered and processed, so it can be used in the filter to obtain an estimated pose. The procedures for the loop section are:

- **Receive sensor and EKF data**: The system design permits more than four inputs, but only four were used in this dissertation as shown in Fig. 1.3. The data that is received by the PF comes from odometry, an EKF, an IPS and a laserscanner. The odometry and EKF inputs provide pose data ($[x\ y\ \theta]$), the IPS provides position data ($[x\ y]$) and the laserscanner gives a matrix L containing the scanned distances and their respective angles relatively to the laserscanner frame with a number of points (k)

dependent on the sensor resolution.

$$L = \begin{bmatrix} d_1 & d_2 & \cdots & d_k \\ \alpha_1 & \alpha_2 & \cdots & \alpha_k \end{bmatrix} \tag{4.1}$$

The configurations of the EKF input used in the PF, are in Section 3.2. While the laser scan data enters directly in the filter, the pose and position data, from other sources, is primarily processed in the Sensor Handler module.

- **Sensor handler**: When pose (odometry, EKF) or position (IPS) data is received, it is processed to be later used in the PF. If position data is received, it is labeled as global positioning data and this received position is saved to be used in the PF, however, if pose data is received, this data is not labeled and the current and previous measurements are transformed into a translation and two rotations ($[u_t, u_{t-1}] \rightarrow [\bar{\delta}_{rot1}, \bar{\delta}_{trans}, \bar{\delta}_{rot2}]$), as shown in Fig. 3.2 and (3.2). Until one of the displacement values (Δd or $\Delta\theta$) has been reached, the transformed data is accumulated each time a new measurement is received as follows:

$$\begin{cases} \delta_{rot1} = \delta_{rot1} + \bar{\delta}_{rot1} \\ \delta_{trans} = \delta_{trans} + \bar{\delta}_{trans} \\ \delta_{rot2} = \delta_{rot2} + \bar{\delta}_{rot2} \end{cases} \tag{4.2}$$

Whenever $(\delta_{rot1} + \delta_{rot2}) > \Delta\theta$ or $\delta_{trans} > \Delta d$, the sensor data is applied to the PF. After being used, the sensor data is reseted. The algorithm of the sensor handler is described in Fig. 4.2.

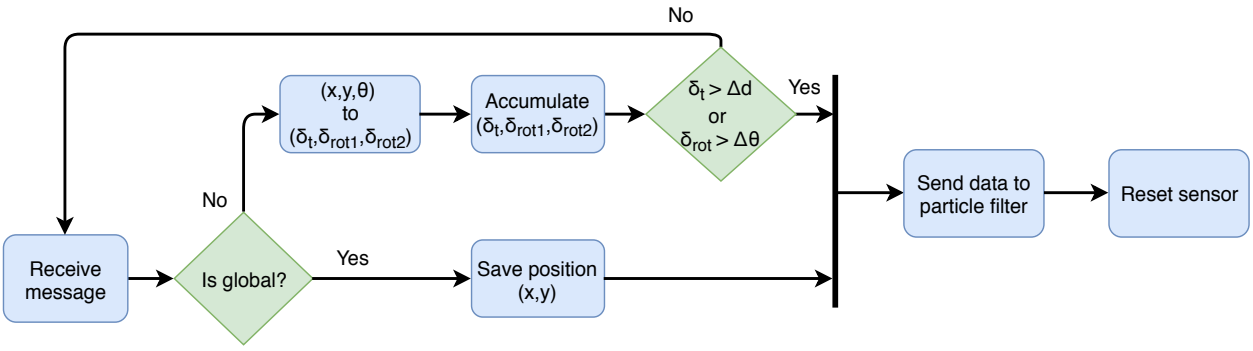


Figure 4.2: Sensor handler steps, where $\delta_t = \delta_{trans}$ and $\delta_{rot} = \delta_{rot1} + \delta_{rot2}$.

- **Particle Filter**: The implemented PF follows the KLD method, described in Subsection 3.1.5, and therefore the order of operations, is slightly different from the one

shown in Fig. 3.1. In this algorithm, the resampling step is the first to run, followed by the prediction step and the last one is the update. The main modifications presented by this PF-based localization approach are in the prediction step where it is possible to receive information from multiple sensors and fuse this information to predict the next pose of the particles. The structure of the filter is shown in Fig. 4.3 and the steps are:

- **Particles initialization**: This step only runs once, during the initialization of the filter. Usually, the particles are randomly created throughout the whole environment. But, in the experiments carried out in this work, the particles were created on the origin of the axis or in an initial pose given by the user since it can be ensured that the platform starts at the pre-defined position. The initialized number of particles is the maximum number of particles (n_{max}), and the weight of each particle is $\frac{1}{n_{max}}$ because at this early stage, they have not yet been validated.
- **Resampling**: At this stage the particles are first ordered from the particle with the lowest to the highest weight. Then, each of the ordered particles receive a number, which contains the summed weight of all particles until that particle. After this, the KLD algorithm is applied, so, from this rearranged set of particles, each time a particle is sampled, it goes through the prediction and the update stages, until the number of processed particles is equal to n_{max}, or the KLD condition is reached. The particles are sampled according to the Multinomial Resampling method, presented in Subsection 3.1.5. When these processes reach to an end, the particle weights are normalized.
- **Prediction**: Before using the sensor information, the algorithm searches for the types of information available and depending on that, a different method of fusion is applied. In the approach implemented here, the types of information are applied according to the method "$1-\alpha$" described in [35]. If pose data is used, the motion model described in Subsection 3.1.2 is applied and if position data is used, the algorithm applies one of the methods in Subsection 4.2.4.
- **Update**: The update stage computes the weight ($w_t = p(z_t|x_t)$) for a particle by using the tracing algorithm described in Subsection 3.1.3. To avoid redundancy during the validation cycle and not to waste processing resources, from the laser points, only some of those are used to acquire the new weight of each particle. The number os points used is defined by the user but they must be equally distant

from each other in terms of number of points.

– **Pose estimation**: After the new particles are calculated, the filter, gets a new estimation of the pose according to the following expressions [48]:

$$\hat{\mathrm{x}}_t^R = \begin{cases} \hat{x}_t^R = \sum_{n=1}^{N} x_t^{[n]} \cdot w_t^{[n]} \\ \hat{y}_t^R = \sum_{n=1}^{N} y_t^{[n]} \cdot w_t^{[n]} \\ \hat{\theta}_t^R = \mathrm{atan2}(\sum_{n=1}^{N} \sin(\theta_t^{[n]}) \cdot w_t^{[n]}, \ \sum_{n=1}^{N} \cos(\theta_t^{[n]}) \cdot w_t^{[n]}) \end{cases} \tag{4.3}$$

with N being the number of used particles and $\hat{\mathrm{x}}_t^R$ the estimated robot's pose

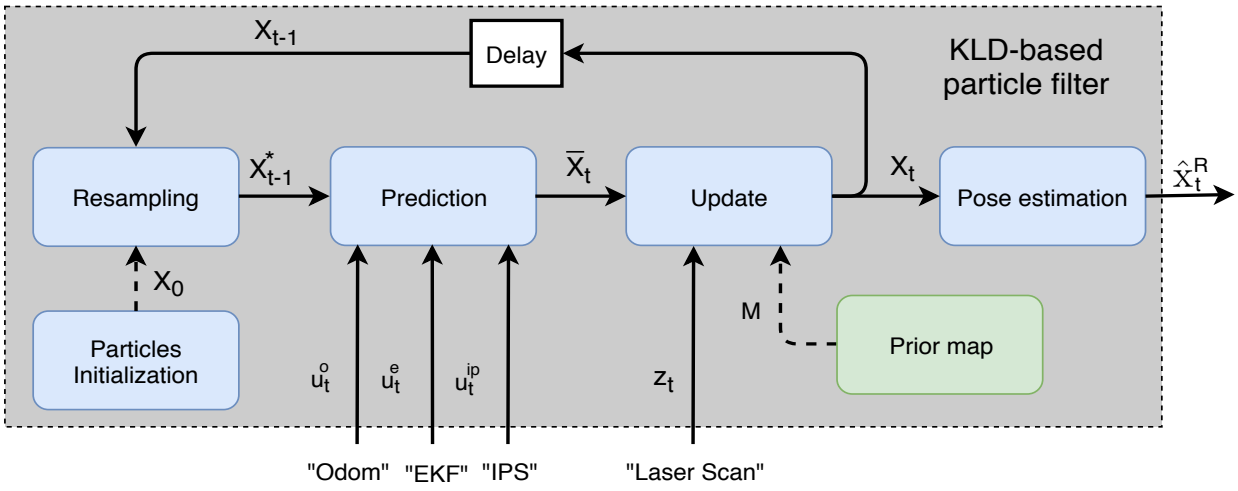


Figure 4.3: KLD-based particle filter block scheme with: X_t and X_{t-1} being the actual and the previous particle sets; X^*_{t-1} the rearranged previous particle set; $\bar{X}_t$ the actual predicted set; X_0 the initial set; $\hat{\mathrm{x}}_t^R$ the actual estimated pose; M the prior map; z_t the observation; u_t (u_t^o,u_t^e,u_t^{ip}) the controls. Odom is an abbreviation for odometry.

4.2 KLD-based particle filter algorithm

This section presents a detailed description of the proposed KLD-based PF algorithm, which is based on the KLD-sampling method [30] presented in subsection 3.1.4. The PF algorithm is usually divided in three stages, and although it is common to apply the resampling stage as the last step of the algorithm, the KLD-sampling algorithm uses the resampling as its first stage and then maintains the order of the remaining steps as shown in Algorithm 3.

Algorithm 3: KLD-based particle filter algorithm with sensor fusion.

Data: Posterior particle set $X_{t-1} = \{(\mathrm{x}_{t-1}^{[n]}, w_{t-1}^{[n]}) | n = 1, ..., N\}$, representing belief $bel(\mathrm{x}_{t-1})$, control measurement $u_t = [u_t^o \ u_t^e \ u_t^{ip}]$, observation z_t , bounds ε and δ, bin size, minimum and maximum number of samples n_{min},n_{max}

1

2 $X_t = \emptyset$, $n = 1$, $n_\chi = 0$, $k = 0$, $op = 0$, $d1 = 0$, $d2 = 0$, $\alpha = 0$; // Initialize

3 $X_{t-1}^* = \mathrm{sort}(X_{t-1})$; // Orders the particles from the lowest to the biggest weight

4 $X_{t-1}^* = \mathrm{prepare}(X_{t-1}^*)$; // Gives the particle set the wc(weight counter)

5 op = $choose(u_t)$; // Choses a fusion option

6 **do**

7 $\quad \mathrm{x}_t^{[n]} = \mathrm{pick}(X_{t-1}^*)$; // Samples a particle from the previous set

8 $\quad \mathrm{x}_t^{[n]} = \mathrm{Prediction}(u_t, \ op, \ \mathrm{x}_t^{[n]})$; // Predicts next state with sensor fusion

9 $\quad w_t^{[n]} = \mathrm{Update}(\mathrm{x}_t^{[n]}, z_t, \ M)$; // Computes Importance weight

10 $\quad \alpha = \alpha + w_t^{[n]}$; // Updates normalization factor

11 $\quad$ **if** $(inEmptyBin(\mathrm{x}_t^{[n]}))$ **then**

12 $\quad\quad k = k + 1$; // Updates number of non-empty bins

13 $\quad\quad setBin(\mathrm{x}_t^{[n]})$; // Marks bin as non-empty

14 $\quad\quad n_\chi = \frac{k-1}{2\epsilon}(1 - \frac{2}{9(k-1)} + \sqrt{\frac{2}{9(k-1)}} z_{1-\delta})^3$; // Updates number of samples

15 $\quad$ **end**

16 $\quad n = n + 1$;

17 $\quad$ **if** $(n >= n_{max})$ **then**

18 $\quad\quad$ break;

19 $\quad$ **end**

20 **while** $(n < n_\chi \ and \ n < n_{min})$;

21 **for** $n := 1, ..., N$ **do**

22 $\quad w_t^{[n]} = w_t^{[n]} / \alpha$; // Normalizes the weight of each particle

23 **end**

24 **return** X_t

4.2.1 Resampling

The method used in this filter is based on the multinomial resampling approach [44] because, it is a simple method to use [42, 43, 44] and it is the same method used in the "amcl" ROS [8] package. Although major part of the sampled particles, have a good weight, there are some drawn particles with a not ideal weight.

The implemented resampling algorithm is divided in the following steps:

- **Sort**: In line 3 (Algorithm 3), the posterior particle set, X_{t-1}, is ordered from the particle with the lowest weight to the one with the biggest weight, giving origin to the sorted set, X^*_{t-1}.

- **Preparation**: In line 4 (Algorithm 3), each particle $\mathrm{x}^{*[n]}_{t-1}$ gets attributed with a new element, $f(\mathrm{x}^{*[n]}_{t-1}) = \sum_{j=1}^{n} w^{[j]}_{t-1}$, with $0 < f(\mathrm{x}^{*[n]}_{t-1}) \leq 1$. These new elements are used to created a new point mass distribution, which will be used to sample a particle from the previous set.

- **Pick**: In line 7 (Algorithm 3), a number is sampled from a uniform distribution ($r \sim \mathcal{U}(0, 1)$) and the algorithm searches for the first number in the point mass distribution whose value is greater than the sampled number. Based on the found number, the particle is picked. This step needs to be inside the loop, because each time a particle is sampled and processed, there is a need to verify if the number of desired samples was achieved.

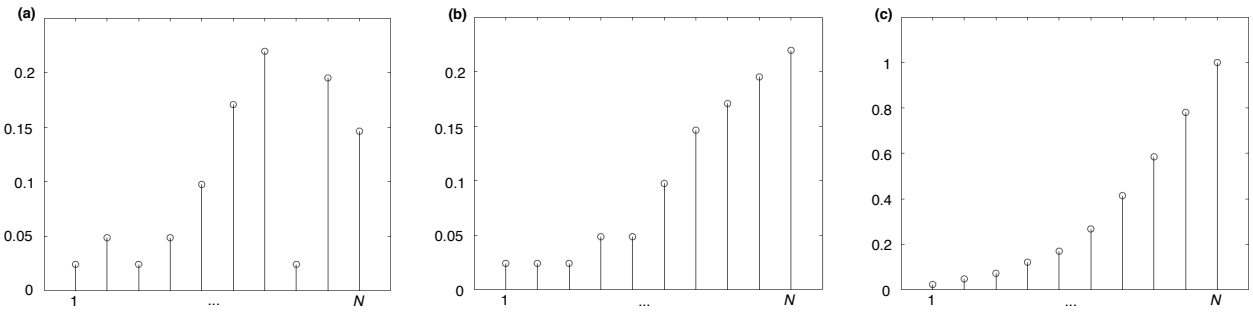


Figure 4.4: Graphics representing an example of the evolution of the resample stage: (a) Previous particle set, X_{t-1}; (b) Sorted particle set, X^*_{t-1}; (c) Point mass distribution, $f(\mathrm{x}^{*[n]}_{t-1}) = \sum_{j=1}^{n} w^{[j]}_{t-1}$, used to sample a particle for the new particle set. With N being the number os particles in the set and $0 < n \leq 1$.

After all particles are sampled for the resampled set, their weights are normalized and the sum of the normalized particle weights is always equal to one ($\sum_{n=1}^{N} w^{[n]} = 1$). Figure 4.4 shows the evolution of the resampling steps.

4.2.2 Prediction

The prediction algorithm is composed by two main tasks: the verification of available sensors and the application of that sensor data. The first task consists in defining the operation mode, which will define the prediction method to use in the picked particle. This is illustrated in Fig. 4.5.

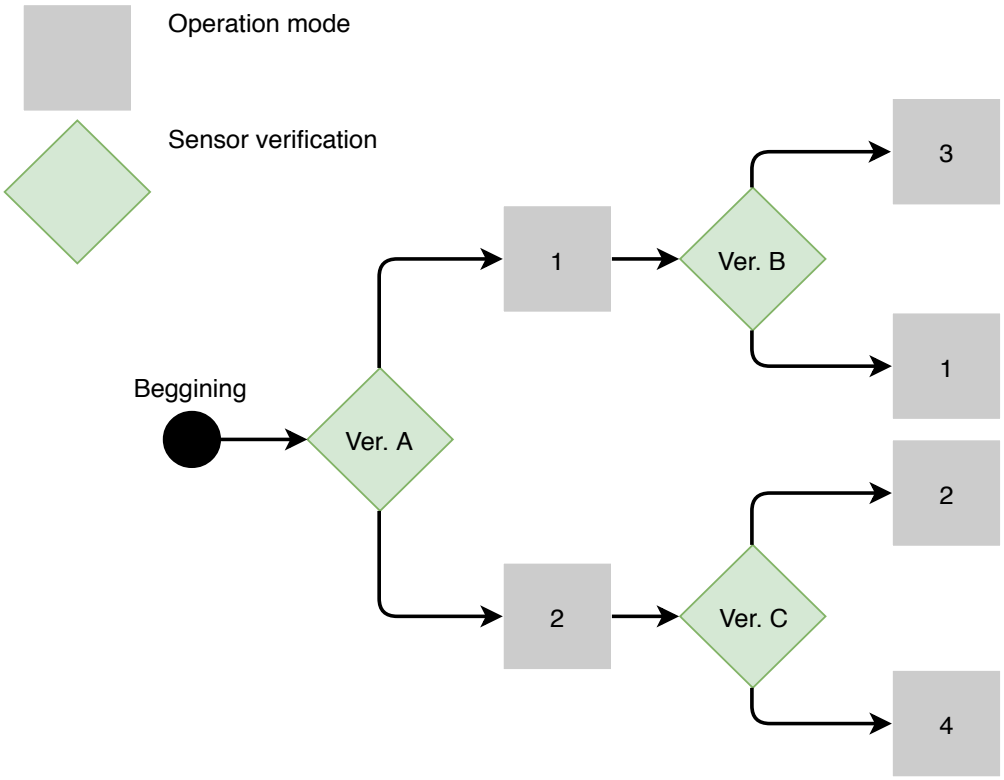


Figure 4.5: Verification of available sensor data through a cascade system.

The verification process (Fig. 4.5), which searches for the available sensor data, is composed by the following steps:

- **Verification A:** This first verification searches for odometry or EKF data. If odometry data is found, the operation mode is set to 1, but if EKF data is available the operation mode is set to 2. If, for some reason, both odometry and EKF data are available, the EKF data is prioritized and the operation mode is set to 2.
- **Verification B:** The process only makes this verification if the operation mode was

priorly set to 1. The system searches for IPS data, and if this data is found the operation mode changes from 1 to 3; if not, the mode stays the same.

- **Verification C:** This verification is only made if the operation mode was priorly set to 2. The system searches for IPS data, and if this data is found the operation mode changes from 2 to 4; if not, the mode stays the same.

It is also important to mention that operation modes 3 and 4 can respectively return to 1 and 2 because the IPS has a low measurement rate and this prevents the localization method from updating at a slower rate. Table 4.1 contains the possible operation modes according to the detected sensor information.

Operation mode	**Sensor data**
1	Odometry
2	EKF
3	Odometry + IPS
4	EKF + IPS

Table 4.1: Table containing the operation mode possibilities and the respective available sensor information.

After the defining the operation mode, the algorithm applies the prediction method, based on the available sensor data. This method was not designed to function without pose information, so there needs to be available at least one sensor that estimates the robot's pose.

Figure 4.6 represents the system that selects the prediction method to be used, based on the defined operation mode. Depending on the number of available sensors, the method is applied accordingly, and in some cases there is more than one type of sensor fusion that can be applied with the same sensors. Aside from the processes of sensor verification and decision making, the prediction algorithm is essentially composed by two functions: the "Motion model" (Algorithm 4) and the "IPS sample" ((4.4)). These functions are applied to pose and position information respectively. When the operation mode is either 1 or 2, the available pose information serves as an input for the "Motion model" function, which applies this information to a particle, as it can be seen in the Algorithm 4.

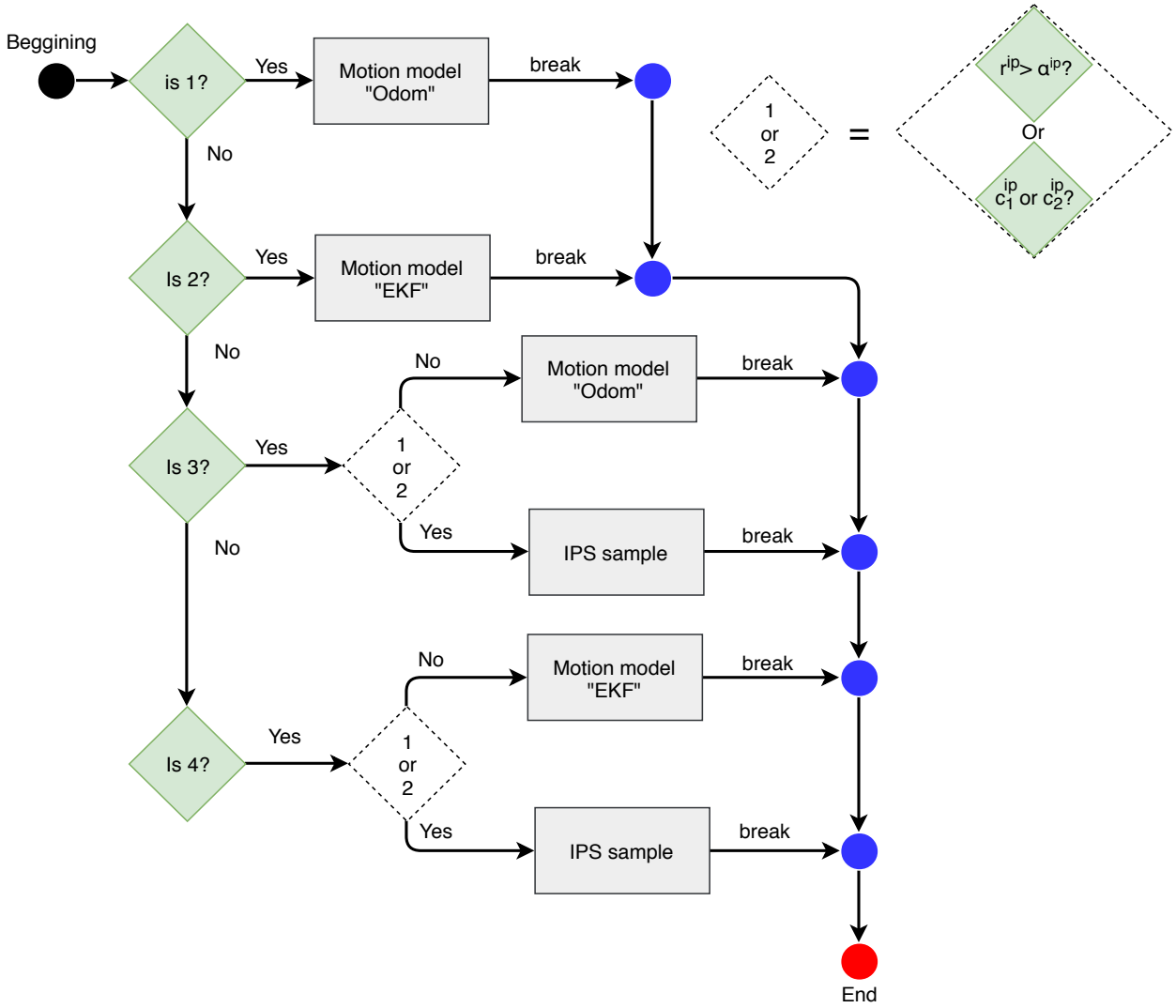


Figure 4.6: Diagram describing the applied prediction methods based on the available sensor data.

If the operation mode is equal to 3 or 4, the algorithm makes an evaluation to select if the next particle is predicted using pose or position information. In the case of using pose information, it's applied the same function as in options 1 and 2. If the position information is used, the "IPS sampling" function, generates a new particle, instead of moving the picked particle, by means of the following equations:

$$\mathrm{x}_t^{[n]} = \begin{cases} x_t^{[n]} = x^{ip} - x^{ge} \sim \mathcal{N}(0, \sigma_x^2) \\ y_t^{[n]} = y^{ip} - y^{ge} \sim \mathcal{N}(0, \sigma_y^2) \\ \theta_t^{[n]} = \hat{\theta}_{t-1}^{R} - \theta^{ge} \sim \mathcal{N}(0, \sigma_\theta^2) \end{cases} \tag{4.4}$$

where $x_t^{[n]}$, $y_t^{[n]}$ and $\theta_t^{[n]}$ are generated by applying sampled errors from normal distributions, to the position given by the sensor (x^{ip} and y^{ip}) and the last orientation estimated by the PF ($\hat{\theta}_{t-1}^{R}$). The standard deviations σ_x^2, σ_y^2 and σ_θ^2 define the maximum error that can

respectively be applied to $x_t^{[n]}$, $y_t^{[n]}$ and $\theta_t^{[n]}$. The sampled errors applied to $x_t^{[n]}$, $y_t^{[n]}$ and $\theta_t^{[n]}$ are respectively x^{ge}, y^{ge} and θ^{ge}. The IPS methods used in operation modes 3 and 4, to select which function to be used, are described in Subscetion 4.2.4.

Algorithm 4: Motion model algorithm adapted from [7].

Data: Particle $\mathrm{x}_t^{[n]} = \{x_t^{[n]}, y_t^{[n]}, \theta_t^{[n]}\}$ and motion parameters $\{\delta_{rot1}, \delta_{trans}, \delta_{rot2}\}$

1

2 $\hat{\delta}_{rot1} = 0$, $\hat{\delta}_{trans} = 0$, $\hat{\delta}_{rot2} = 0$; // Initialize

3

4 $\hat{\delta}_{rot1} = \delta_{rot1} - \delta \sim \mathcal{N}(0, (\alpha_1 \cdot \delta_{rot1}^2 + \alpha_2 \cdot \delta_{trans}^2))$;

5 $\hat{\delta}_{trans} = \delta_{tans} - \delta \sim \mathcal{N}(0, (\alpha_4 \cdot (\delta_{rot2}^2 + \delta_{rot1}^2) + \alpha_3 \cdot \delta_{trans}^2))$;

6 $\hat{\delta}_{rot2} = \delta_{rot2} - \delta \sim \mathcal{N}(0, (\alpha_1 \cdot \delta_{rot2}^2 + \alpha_2 \cdot \delta_{trans}^2))$;

7 $x_t^{[n]} = x_{t-1}^{[n]} + \hat{\delta}_{trans} \cos(\theta_{t-1}^{[n]} + \hat{\delta}_{rot1})$;

8 $y_t^{[n]} = y_{t-1}^{[n]} + \hat{\delta}_{trans} \sin(\theta_{t-1}^{[n]} + \hat{\delta}_{rot1})$;

9 $\theta_t^{[n]} = \theta_{t-1}^{[n]} + \hat{\delta}_{rot1} + \hat{\delta}_{rot2}$;

10 **return** $\mathrm{x}_t^{[n]} = \{x_t^{[n]}, y_t^{[n]}, \theta_t^{[n]}\}$

4.2.3 Update

By using the the prior map (M) and the laser scan measure (z_t) the predicted particle is evaluated in order to determine the particle weight. The process of the update step is described in Algorithm 5. The algorithm computes the particle weight by calculating the difference between the measured scan points and the traced points on the map. The first step is to transform the measure z_t to the particle pose. The transformed scanned points will be evaluated individually, but to optimize the processing time for each particle, not all of the points are verified. Instead, the checked points are separated by a constant number (c_u). By using the tracing method described in Subsection 3.1.3, it is calculated the closest occupied cell point (x_{traced}, y_{traced}) in the line between the particle and the measured point (x_{hit}, y_{hit}). The weight is obtained by applying the difference between the measured point and the closest occupied cell point to the normal distributions $\mathcal{N}_x$ and $\mathcal{N}_y$, which are respectively referent to the x-axis and y-axis.

Algorithm 5: Update stage algorithm.

Data: Particle $\mathrm{x}_t^{[n]} = \{x_t^{[n]}\ y_t^{[n]}\ \theta_t^{[n]}\}$, laser scan $z_t = \{z_1, ..., z_K\}$ wiht n being the number of ranges of the lase scan and c_u a constant

$\alpha = \alpha_{min}$; // Minimum angle of the laser scan range

$i = \alpha_{inc}$; // Angle increment between measurements, z_k

$w_t^{[n]} = 0$, $x_{hit} = 0$, $y_{hit} = 0$, $j = 0$;

$z_t = trans(z_t, x_t^{[n]}, y_t^{[n]}, \theta_t^{[n]})$; // Transforms the scan to the position of the particle

for $k = 1$; $k < K$; $k = k + c_u$ **do**

  $x_{hit}^k = x_t^{[n]} + z_k cos(\alpha)$;

  $y_{hit}^k = y_t^{[n]} + z_k sin(\alpha)$;

  $(x_{traced}^k, y_{traced}^k) = nearest(x_t^{[n]}, y_t^{[n]}, x_{hit}^k, y_{hit}^k)$; // Tracing algorithm

  $w_t^{[n]} = w_t^{[n]} + \mathcal{N}_x(0, x_{traced}^k - x_{hit}^k) \cdot \mathcal{N}_y(0, y_{traced}^k - y_{hit}^k)$;

  $\alpha = \alpha + c_u \cdot i$;

  $j = j + 1$;

end

$w_t^{[n]} = \frac{w_t^{[n]}}{j}$;

return $w_t^{[n]}$;

4.2.4 IPS methods

This section presents two decision-making approaches used to decide between using pose or position data. One of the methods is based on the method presented in [35], while the other is based on the verification of conditions related to the distance between points or time between received measurements. The block that represents this decision-making process is shown in Fig. 4.7.

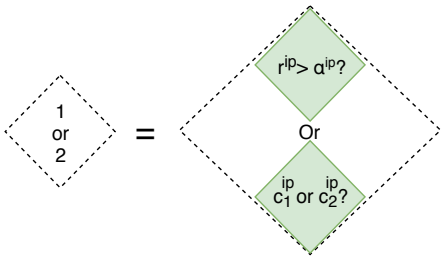


Figure 4.7: Decision block representing two approaches used in the particle filter to decide between the use of pose or position data.

The first method (IPS1) generates a random number between 0 and 1 ($r^{ip} \sim \mathcal{U}(0,1)$) and based on that number, it uses the pose or position data, as it follows:

$$\begin{cases} \text{“}IPS\ sample\text{”}\ \textbf{if}\ \ r^{ip} > \alpha^{ip} \\ \text{“}Motion\ model\text{”}\ \textbf{otherwise} \end{cases}$$

with α being the constant that represents the percentage of particles that use pose data. Conversely, the other approach (IPS2) is based on a method where one of two conditions (c_1^{ip} or c_2^{ip}) needs to be verified in order to use position data. One condition (c_1^{ip}) is verified through the distances between the position control measure (u_t^{ip}) and 2 other points, which are: the last position control measure (u_{t-1}^{ip}) and the picked particle ($\mathrm{x}^{[n]}$). If the distance between the picked particle and the control measurement (d_1) is lower than a tolerance threshold (th^{ip}) or if the distance amidst the actual measure and the posterior measure (d_2) is inferior to constant distance error (d_{err}), the first condition (c_1^{ip}) is met. Figure 4.8 shows an example of the distances between mentioned points.

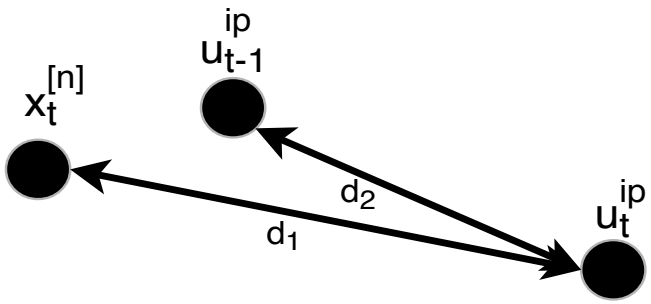


Figure 4.8: Example of distance calculation for the verification of one condition for the method.

The second condition (c_2^{ip}) is verified through the time passed between measurements. If the elapsed time since the last measure (Δt) is higher than a defined time threshold (T_{th}), it is used the position information. Having the distances calculated and the time between measurements, the conditions are verified as follows:

$$\begin{cases} \text{“}IPS\ sample\text{”}\ \textbf{if}\ (d_1 < th^{ip}\ and\ d_2 < d_{err})\ \textbf{or}\ \Delta t > T_{th} \\ \text{“}Motion\ model\text{”}\ \ \textbf{otherwise} \end{cases}$$

Chapter 5

Validation platform

The mobile robot represented in Fig. 5.1 is a prototype for an AGV that was developed in ISR-UC. In its final stage, it should be able to construct a map of an unknown environment and localize itself in said environment, but in this dissertation the objective is to obtain the pose estimation of the mobile robot using sensor information and a prior map. This chapter provides a brief overview of the physical setup of the AGV mobile robot and its hardware.

Figure 5.1: Platform is composed by: (1) Processing Unit; (2) Marvelmind Beacon; (3) Gamepad controller; (4) IMU sensor; (5) Lead-Acid Batteries; (6) Wheels with encoders; (7) Laser Scanner; (8) RoboteQ Motors Controller.

5.1 Platform kinematics

The validation platform is composed by two standard fixed wheels (L and R) and three spherical wheels (s_1, s_2 and s_3) for support. Having this configuration, the platform's kinematics is only governed by the standard wheels, so this platform is considered a differential drive platform as it is shown in Fig. 5.2 (adapted from [38]). The distance between standard wheels baseline is defined as d_w.

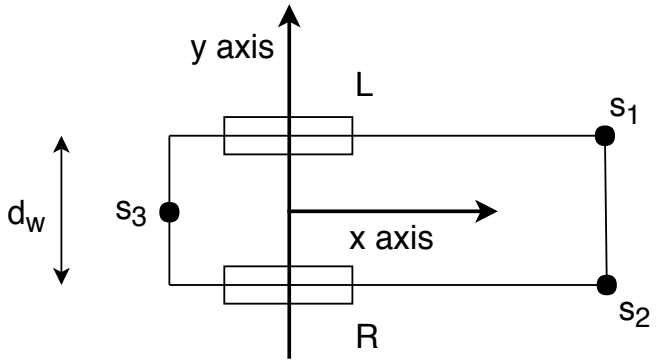


Figure 5.2: Representation of the AGVs motion model.

5.2 Hardware architecture

The AGVs system is composed by a low-level part related to the power and mechanical components, and a high-level part used for processing data. Each hardware module that composes the AGV is described below.

5.2.1 Processing unit

A laptop located in the AGV platform, runs software in the ROS environment [8] to generate speed commands, that are sent to the motors, and receives sensor data, which is used in the localization algorithm. Receives as inputs the wheels encoder measurements, laser scan data from the Hokuyo Laser [49], the linear acceleration (x-axis) and angular velocity (z-axis) from the Xsens IMU sensor [50] and the absolute position from the Marvelmind system [51]. The Processing Unit is located in the platform, so that incoming data from the sensors can be obtained and processed with minimum delay. This laptop has 15.6 GB of RAM memory and an Intel Core i7-5500U processor.

5.2.2 Marvelmind system

The commercial system, developed by Marvelmind, its composed by a modem, one or more mobile beacons, and multiple static beacons (minimum of three). This system uses ultrasound and radio signals, with a frequency of 433MHz, to compute the position of the mobile beacons using a trilateration algorithm [17]. It's only possible to use this algorithm because the coordinates of the static beacons are given *a priori* and because the system gives the distances between the mobile beacon and the static beacons. The mobile beacon stays in the AGV (see Fig. 5.1) and its position is referenced in the user-defined coordinates system, where the origin is configured by the user in any of the static beacons placed on the infrastructure. The modem serves as a "communication bridge" between the beacons and the processing unit. This system is capable of computing the position of each beacon without human assistance, which facilitates the setup on industrial environments. Figure 5.3 shows the system's hardware.

(a) Modem. (b) Beacons.

Figure 5.3: Hardware components of the Marvelmind system with: (a) Modem that communicates with all beacons and the processing unit; (b) Beacons used for trilateration with one staying in the AGV and the rest being scattered throughout the environment.

As an example of the configuration made on the Marvelmind software, Fig. 5.4 represents the map of beacons configured in the Marvelmind software. The equivalent blueprint of the floor, in which this beacon's map was configured, is represented in Fig. A.1. Table 5.1 contains the technical specifications of the Marvelmind hardware.

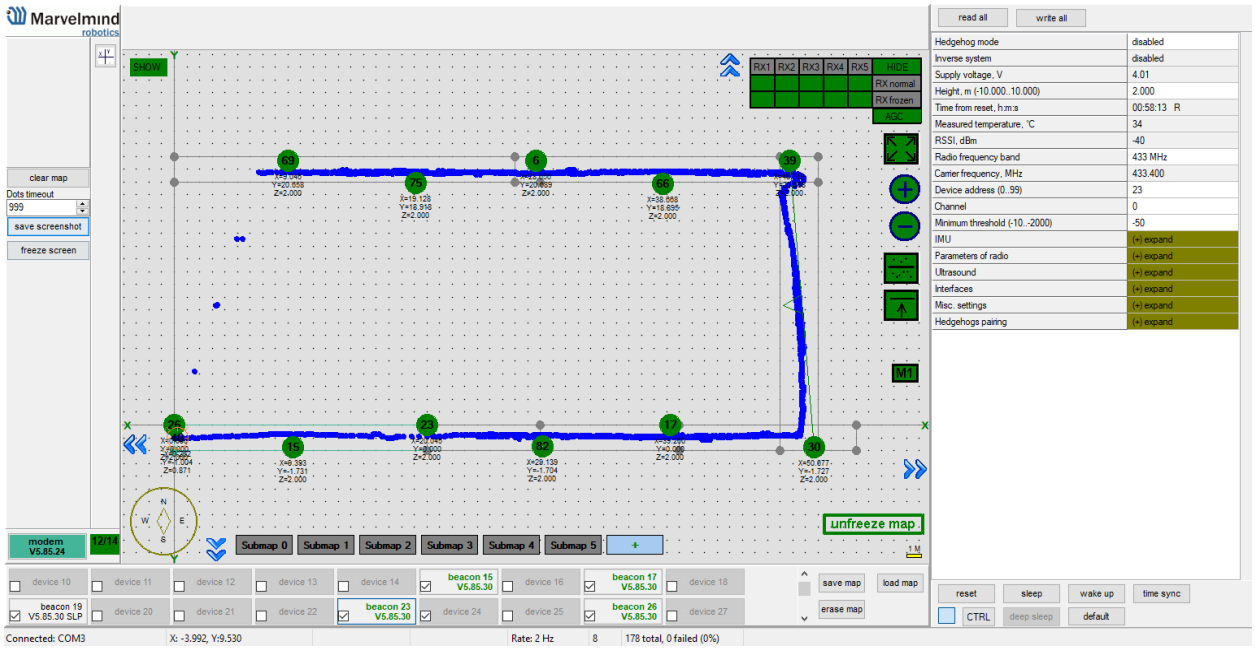


Figure 5.4: ISR-UC floor 0 beacons map configured on the Marvelmind software.

Max distance between beacons	30 m	**Principle**	trilateration
Coverage area	1000 m^2	**Technology**	Ultrasound
Precision error	1-3% of the distance given by the beacon	**Location precision**	Differential precision: $\pm$ 2 cm
Position update rate	0.5 - 45Hz	**Beacon size**	55x55x33 mm

Table 5.1: Technical specifications of the Marvelmind system adapted from [51]

5.2.3 Encoders and RoboteQ motors controller

The wheels encoder resolution is 980 pulses per revolution (np_e) and this information is necessary to obtain the platform pose estimation trough odometry.

The RoboteQ Motors Controller (SDC2130) is used as a PID controller, which receives speed commands from the processing unit and transforms them into voltage and current outputs towards driving one or two DC motors. The transmission of the revolution pulses information is made trough USB communication so that they can be processed into odometry information.

$$D_r = \frac{2\pi \cdot r_w \cdot np_r}{np_e} \quad D_l = \frac{2\pi \cdot r_w \cdot np_l}{np_e} \tag{5.1}$$

$$\Delta d = \frac{Dr + Dl}{2} \quad \Delta\theta = \frac{Dr - Dl}{2} \tag{5.2}$$

$$\begin{cases} x_t^o = x_{t-1}^o + \Delta d \cos(\theta_{t-1}^o + \Delta\theta) \\ y_t^o = y_{t-1}^o + \Delta d \sin(\theta_{t-1}^o + \Delta\theta) \\ \theta_t^o = \theta_{t-1}^o + \Delta\theta \end{cases} \tag{5.3}$$

The odometry equations are represented from (5.1) to (5.3), where: np_r and np_l are respectively the number of pulses detected since the last measure, D_r and D_l are the displacements of the right and left wheels, Δd and $\Delta\theta$ are the linear and angular displacement, r_w is the wheel's radius and x_t^o, y_t^o and θ_t^o are the odometry estimation measurements.

The linear velocity is limited to a maximum of 0.5 [m/s].

5.2.4 Xsens Mti-G IMU sensor

This IMU [50, 52] is located right above the origin of the mobile robot. The frequency of the measurements is of 100Hz and since the measurements are to noisy, the data coming from this sensor serves mainly to compensate flaws of the other sensors, as it is in the case of the Marvelmind system, which does not provide the mobile beacon's orientation. The sensor gives a great amount of information such as linear accelerations, angular velocities and magnetic field values on all xyz-axis, however, for the purposes of this dissertation, only the linear acceleration in x-axis and angular velocity in z-axis were used.

5.2.5 Hokuyo laser scanner

The Laser scanner is the most important sensor on the platform, because the proposed localization algorithm in this dissertation is a laser-based algorithm. As Fig. 5.1 shows, this sensor is displaced from the base of the platform (middle point between the wheels), and in order to use the scan information properly, a rigid transformation is applied to the laser, at the x-axis. The laser is dislocated 0.7m along x-axis, so the necessary transformation matrix, is:

$$^{Robot}T_{Laser} = \begin{bmatrix} 1 & 0 & 0 & 0.7 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The laser scanner used in the tests was the Hokuyo UTM-30LX [49]. It uses laser source (λ = 870nm) to scan a 270º semicircular field. Table 5.2 shows the laser's main specifications. Sensor's measurement data along with the correspondent angle are transmitted through USB communication.

Supply Voltage	12V DC +- 10%	**Angular Resolution**	0.25º
Guaranteed Range	0.1 $\sim$ 30m	**Measurement Step**	1080
Maximum Range	0.1 $\sim$ 60m	**Scan Speed**	25ms
Scan Angle	270º	**Measurement Resolution**	1mm

Table 5.2: Hokuyo's UTM-30LX laser main specifications [49].

5.2.6 Gamepad controller

The gamepad controller is used as a HRI (Human Robot Interface). It sends velocity commands to the processing unit, which is going to send them again to the motor's PID controller (RoboteQ Motors Controller). The left joystick gives linear velocity, and the right one gives angular velocity commands.

5.3 Sensor data

Various sensors were incorporated in the mobile robot. Their raw sensor data is received by the processing unit and turned into viable data for the localization method. On Chapter 6, an overview of how the sensors are being used to test the localization methods is presented. Each sensor has its own measurement frequency, and this also depends on the computational time that takes for the raw data to be processed. It is important to mention that the MSPF[2] update rate varies from 40ms to 100ms. Table 5.3 shows each sensor frequency.

Sensors	**Measurement frequency**
Odometry	20 Hz
IMU	100 Hz
Indoor GPS	4-8 Hz
Laser Scanner	40 Hz

Table 5.3: Sensors computational times.

Chapter 6

Experimental validation

In this chapter the main experimental results from different scenarios and correspondent analysis are presented. The experiments include:

1. The comparison between the MSPF^2 and the AMCL methods in the same scenario.
2. Tuning of the MSPF^2 software parameters.
3. Testing the different approaches in two environments.
4. Evaluating the performance of some approaches on a localization loss scenario.

The tests were performed in the ROS environment [8], on two different structures: the first containing the developed MSPF^2 package, and the other, the modified AMCL package, being respectively represented in Fig. 6.1 and Fig. 6.2.

The AMCL package was used as benchmark, thus the results can be compared with the ones from the MSPF^2 package. Although the idea is to compare the developed package with a consolidated one, the AMCL was modified so it could receive messages from either the "odom" (odometry), or the "robot_pose_ekf" (EKF) ROS topics.

The EKF package mentioned in this dissertation was not tuned because it was only used as a different source of motion data. This motion data is a result of filtering odometry or odometry fused with IMU data.

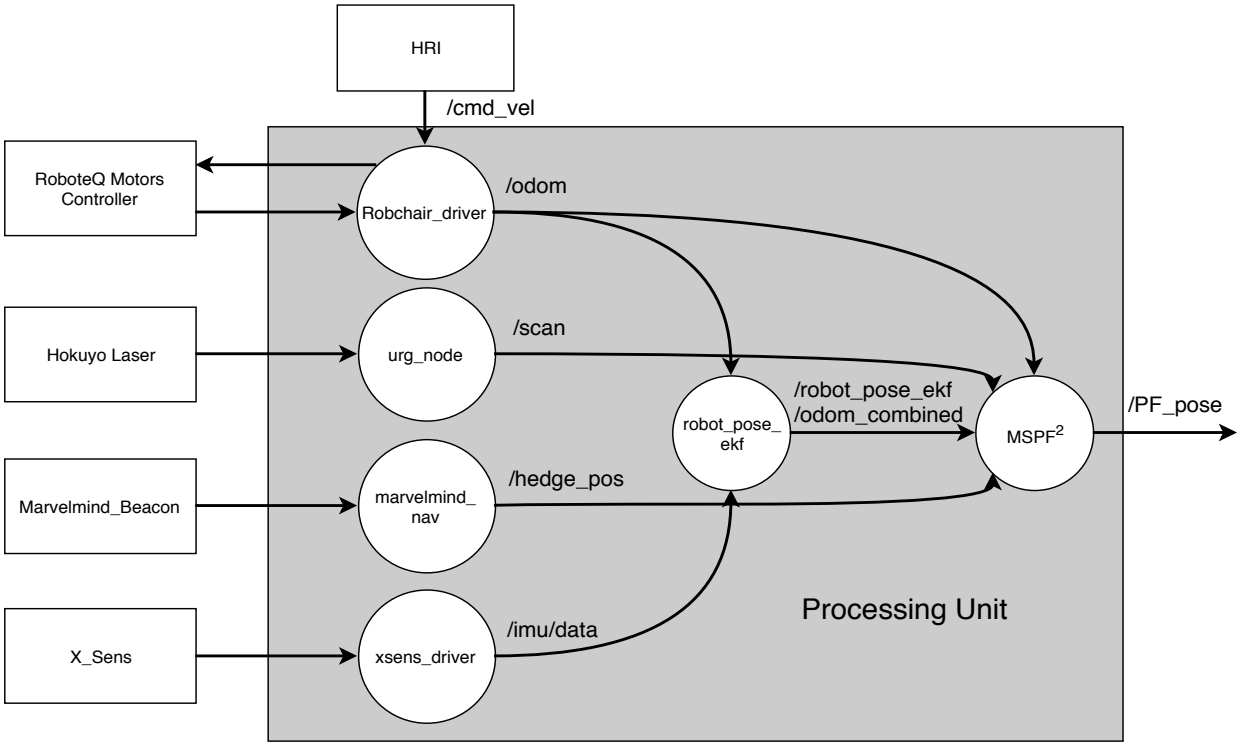


Figure 6.1: ROS schematic for the MSPF^2 localization method, which contains the ROS nodes used in the system, the subscribed and published topics, and the hardware that communicates with mentioned nodes.

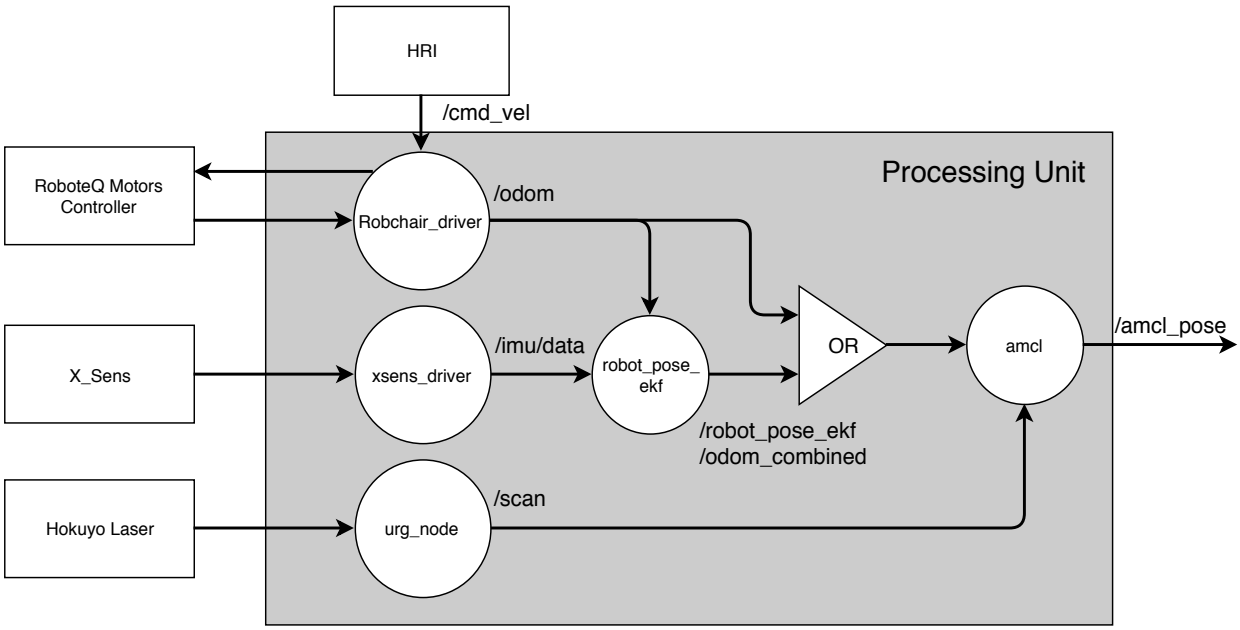


Figure 6.2: ROS schematic for the modified AMCL method, which contains the ROS nodes used in the system, the subscribed and published topics, and the hardware that communicates with mentioned nodes.

6.1 Methods comparison

The first experiment consisted in comparing the two systems on the same conditions. For this purpose, both were tested on the same scenario with equal parameters as follows: $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 0.04$; $\Delta d \approx 0.2m$; $\Delta\theta \approx 0.523rad$; $n_{max} = 2000$; $n_{min} = 100$. The parameters related to error sampling $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ serve to apply more or less error to the predicted particle positions, as explained in Section 3.1.2, while the parameters Δd and $\Delta\theta$ limit the particles' update rate. Figure 6.3 shows the room where this comparison was made. Both systems use grid cell maps for validation of the predicted particle positions, with the map used for this experiment being represented in Fig. 6.4. The intended trajectory is represented in Fig. 6.4 by the red arrows.

Figure 6.3: Picture of the ISR-UC experiments room.

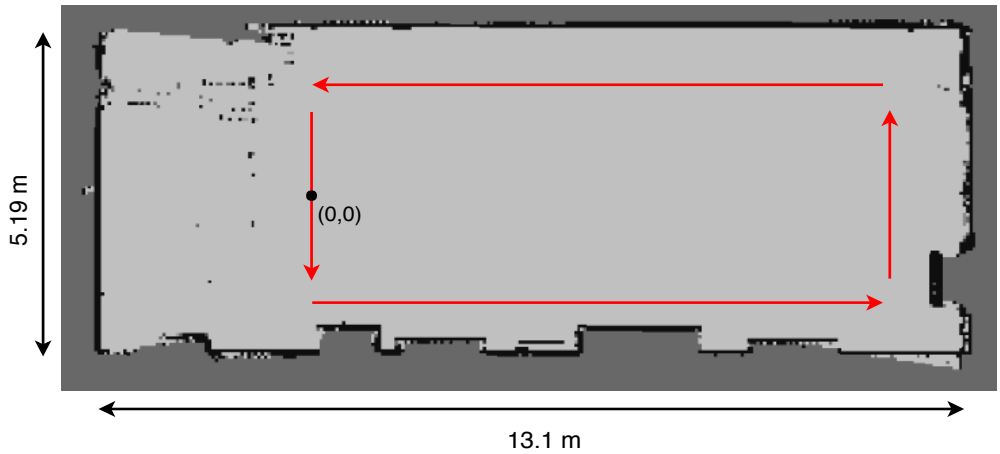


Figure 6.4: Occupancy grid map representing the ISR-UC experiments room with the defined origin of the axis marked as the point (0,0). The intended trajectory is marked by the red arrows.

In order to compare the systems, two metrics were used for evaluation. In both metrics the laser scan is used to estimate a score, but while one scores according to the update method presented in Subsection 4.2.3, the other simply verifies if the cell detected by the scan is occupied or not. In sum, for all points given by the scan ($p_{hit}^{[k]} = [x_{hit}^{[k]}, y_{hit}^{[k]}]$, with $1 \leq k \leq 1080$), the methods evaluate as follows:

- **Metric 1** (M1): $s^{[k]} = \mathcal{N}(0, x_{traced}^{[k]} - x_{hit}^{[k]}) \cdot \mathcal{N}(0, y_{traced}^{[k]} - y_{hit}^{[k]})$, with $0 \leq s^{[k]} \leq 1$
- **Metric 2** (M2): if $p_{hit}^{[k]}$ is occupied, $s^{[k]} = 1$,otherwise $s^{[k]} = 0$

The estimation of localization score is given by the following expression:

$$S = \sum_{k=1}^{1080} s^{[k]}, \ with \ 0 \leq S \leq 1080 \tag{6.1}$$

By using one of ROS tools, the data from the sensors was saved in a rosbag file ($i.e$ ".bag" file), so it could be applied to the different methods. The idea of this test was to drive the robot over a path on the floor marked with magnetic tape, so that the points of the path could serve as a ground truth. The intended path starts at the maps origin point and ends after one complete lap. This path is represented in Fig. 6.5, along with sensors' and EKF data.

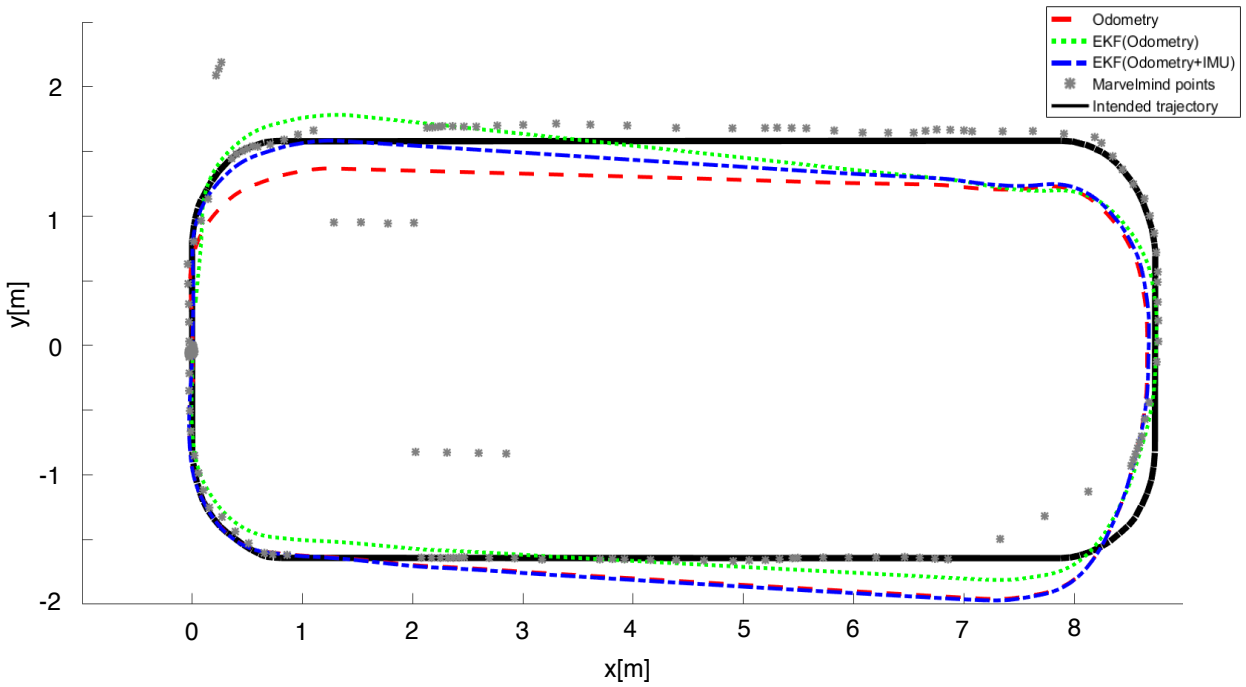
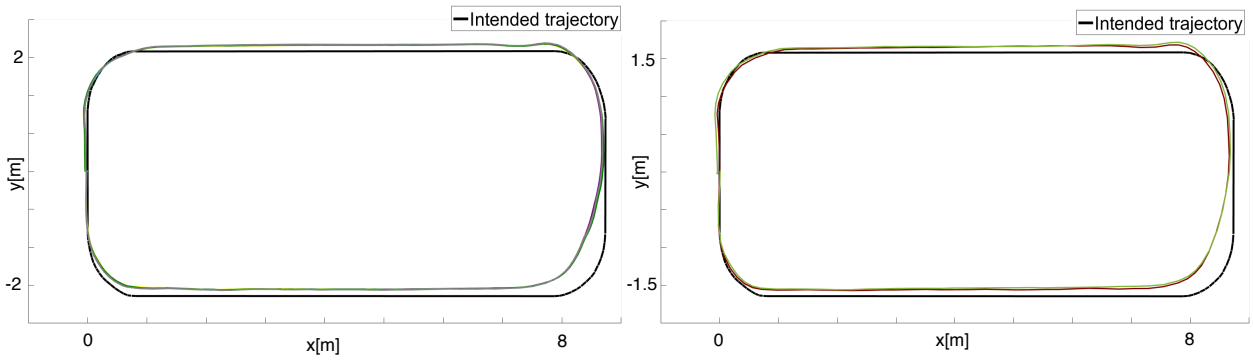


Figure 6.5: Representation of the sensors' and EKF data overlapped with the intended testing trajectory.



(a) MSPF^2 paths.

(b) AMCL paths.

Figure 6.6: Overlap of the trajectories given by the MSPF^2 and AMCL methods with the intended path.

As can be seen in Fig. 6.5, the odometry measurements have some associated error, but this can be attenuated by using the EKF. The use of IMU information on the EKF, adjusts the platforms pose slightly better at the end of the path, since it is the only pose information that ends at the same point as the beginning of the path.

Some of the points given by the beacons have some outliers that could be caused by occlusions or by some internal configuration error. Even so, the majority of the points are close to the intended path. The sensors' and EKF data from Fig. 6.5 was applied to both structures (Fig. 6.1 and Fig. 6.2) according to the pretended fusion methods (all tested methods appear in Table 6.1), and the resulting paths from this tests can be seen in Fig 6.6. From the representation of the paths in Fig. 6.6, both systems appear to give similar trajectories, but presenting some slight variations between methods. The paths estimated by the methods are not entirely equal to the one pretended, but this may be caused by many reasons, such as precision errors from hardware or software and errors originated by human driving. Many conclusions can be drawn from the values of Table 6.1. The evaluations of the sensors' and EKF data show low scores, as it should be expected from what is seen in Fig. 6.5, since major part of the estimations given by the sensors' and EKF data have an incorrect position and orientation. The scores of the PF methods are higher because even if the position is not entirely correct, with the correct orientation, the laser evaluation should give better scores. The second evaluation gives worse scores because it consists of a simple binary evaluation, which discards the scan points that do not coincide with occupied cells. Because of this, the remaining scenario evaluations were only made based on the first evaluation metric.

Methods	**RMS**^{M1}(%)	**Mean**^{M1}(%)	**RMS**^{M2}(%)	**Mean**^{M2}(%)
Odom	19.2079	15.7086	14.2421	11.9188
EKF(Odom)	29.4837	25.4813	19.9629	16.7636
EKF(Odom+IMU)	33.3997	28.2895	24.5289	21.1390
MS(Odom)	73.4608	72.5176	65.4027	63.8518
MS(EKF(Odom))	72.4424	71.6499	62.3197	60.9444
MS(EKF(Odom+IMU))	**74.0880**	**73.1766**	**65.9523**	**64.4609**
MS(Odom+IPS1)	73.7471	**72.7721**	65.2060	63.6005
MS(Odom+IPS2)	**73.7636**	72.7562	**65.8472**	**64.1705**
MS(EKF(Odom)+IPS1)	72.6698	71.8542	62.4718	60.9866
MS(EKF(Odom)+IPS2)	73.1479	72.3105	63.5779	62.2810
MS(EKF(Odom+IMU)+IPS1)	**73.9271**	**72.9962**	**65.4734**	**63.9615**
MS(EKF(Odom+IMU)+IPS2)	72.9778	72.0960	63.5136	62.1472
AMCL(Odom)	70.1248	68.4955	58.3771	56.0754
AMCL(EKF(Odom))	63.7728	62.2694	48.6091	46.1578
AMCL(EKF(Odom+IMU))	69.2302	67.9259	58.5352	56.9268

Table 6.1: Table containing the RMS and the mean of the two different metric evaluations made on tested methods. The values in bold are the three best values of each column.

The AMCL does not increase in performance by using the EKF, but the MSPF^2 does in some methods. Using the Marvelmind system also does not improve much more than the EKF, but it still provides better scores than the methods using just odometry and laser scan data. These conclusions are not enough to determine which of this methods is more reliable, because these tests were made in a small environment on a simple path and with no dynamic obstacles. MS is a short for MSPF^2 that is used in this chapter's tables. Despite these facts, the MSPF^2 appears to show similar and better results than the AMCL. It is also important to emphasize that this scenario evaluations were only made when new poses were estimated due to the structure of the "amcl" code, which did not permit that evaluations were made in the time between estimations. However, since the remaining scenarios were only evaluated with the MSPF^2 system, the time between estimations was also evaluated. Both structures were compared on this scenario because they used the same parameters in the PF packages, but after this scenario, the MSPF^2's parameters were tuned, so the AMCL was no longer tested.

6.2 Parameters tuning

Before proceeding to the next scenarios, the parameters of the MSPF^2 were calibrated in an attempt to increase the performance of the localization methods. The variations upon the parameters were always made on the same scenario, so the results could be comparable. The results were analyzed with the first evaluation method described in Section 6.1. After a parameter evaluation, its value was set as the value that showed better results, so it could be used on the remaining evaluations. The evaluation results of the tuning process are represented in Table 6.2 and the order of evaluations was the same as the order of this table.

Parameter	**Max(%) Mean**	**Min(%) Mean**	**Max(%) RMS**	**Min(%) RMS**	**Best Value**	**Max Value**	**Min Value**
α_1	62.6296	59.8889	65.8576	62.9352	0.0208	0.1	0.001
α_2	62.6296	59.8889	65.8576	62.9352	0.001	0.1	0.001
α_3	62.6296	59.8889	65.8576	62.9352	0.0208	0.1	0.001
α_4	62.6296	59.8889	65.8576	62.9352	0.0802	0.1	0.001
Δd	69.3571	44.9537	71.2161	51.7500	0.05	0.3825	0.05
$\Delta\theta$	69.3571	44.9537	71.2161	51.7500	0.01	0.556	0.01
α^{ip}	70.3889	47.5741	71.5278	53.8796	1.0	1.0	0.0
th^{ip}	70.5741	10.1019	71.8056	21.7870	0.43	0.5	0.0
n_{max}	70.7130	48.2130	71.9167	54.3704	1740	10000	100

Table 6.2: Data from the tuning tests, including the evaluation values made with Metric 1, the experimented ranges and the Best Value for for each parameter. The proposed testing range was between the Min and Max values defined on the table.

In the appendix are shown the RMS for the tested values of each parameter. The parameters α_1, α_2, α_3, α_4 values were altered within the same test, so that all the combinations between the proposed range could be tried. As could be expected, the best performances were obtained when the parameters had low values because when adding too much error to the particles, they start to diverge from the real robot's pose. The Δd and $\Delta\theta$ were in the same test as the prior parameters. In this case, the best value for both parameters was the lowest tested value, since the pose actualization rate is inversely proportional to these parameter's values. The parameters were not tested with lower values due to the cases where the platform is not moving, since it does not need a new pose if it still is in the same

point. The best value for the α^{ip} is 1.0, which means that the position information is never used. This result is not conclusive, because the tested scenario had a simple trajectory, no dynamic obstacles and many features. For these reasons, further evaluations were made with this parameter defined as 0.9, so that position information could help if needed, but without ruining the pose estimation. The th^{ip} best value is 0.5m but this leads to the possibility of particles, that strayed too much from the real pose, too be considered as good particles. Since the results from 0.16m to 0.5m are stabilized, the value defined for the remaining tests was 0.20m. As for the n_{max} parameter, the best value is 1740 particles, which shows that a low number of particles can not be enough for a good pose estimation, but the greater the number, the longer it takes to estimate the pose, thus leading to worse estimation. Figures A.2 to A.6 represent the score's RMS of the tuning tests for the tested parameters.

6.3 Large environment validation

After calibrating the parameters, the next step was to evaluate the behavior of the methods in a larger scale environment containing some zones short on features. This serves to test if the methods can locate the robot in adverse conditions and, in case of getting lost, if it is possible to locate the robot again. For this purpose, the environment chosen was the floor 0 of the DEEC-UC building.

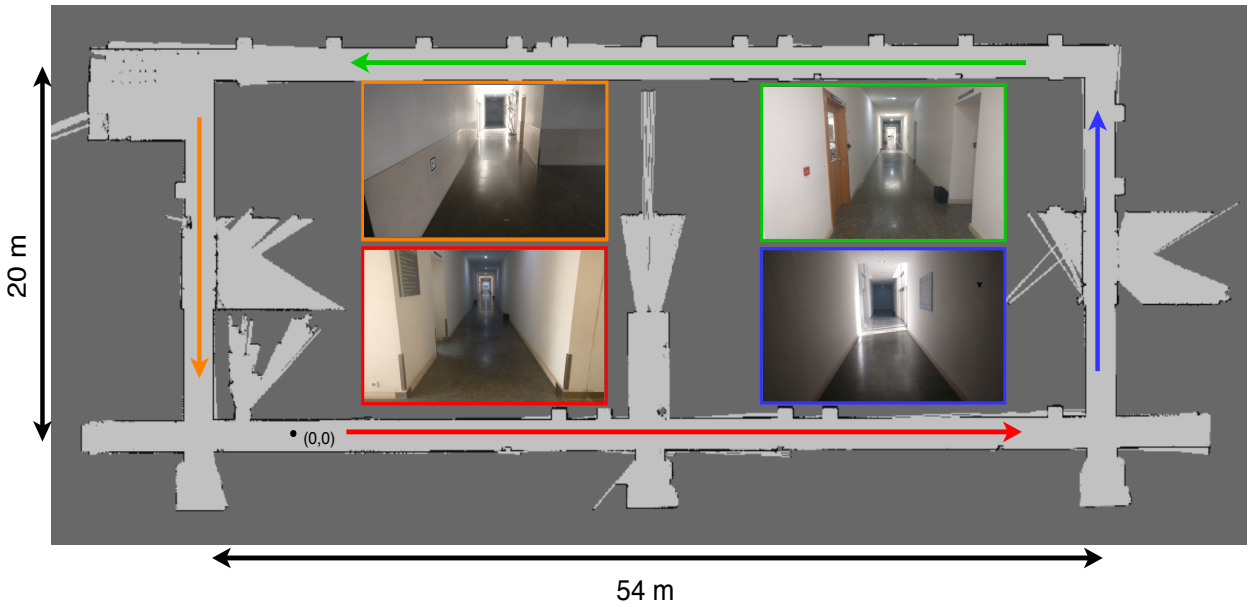


Figure 6.7: Occupancy grid map representing the floor 0 of the DEEC-UC with the defined origin of the axis marked as the point (0,0). The pictures are referent to the arrows with the same color as its contour (*e.g.*, red arrow refers to the picture with the red contour).

6.3.1 Scenario 1

This scenario implicates going around the floor twice, with the first lap being counter-clockwise and the second lap being in clockwise direction. The initial position is the origin of the axis defined in Fig. 6.7.

Figure 6.8 represents the overlap of the trajectories obtained with each of the different used methods. At a larger scale the variations between the methods are not noticeable, except for the corners of the trajectories. In the zoomed "Area 1", some slight variations can be noticed in one of the curves, more precisely in the more abrupt curve. This shows that the methods diverge when angle variations are too abrupt. Nonetheless, when the platform movement stabilizes, the methods begin to converge. Area 2 is referent to the zone of the initial and final points of the trajectories and, as it can be seen, the variations are very low, considering the zoomed scale.

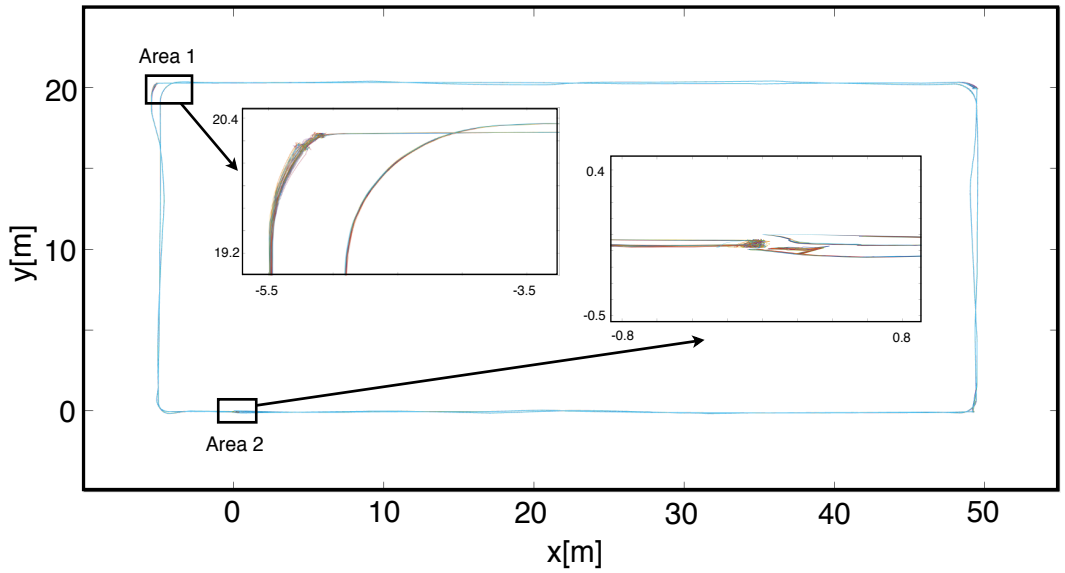


Figure 6.8: Representation of the overlapped trajectories from 10 samples of all the MSPF^2 methods. Area 1 is a zoom of a curve while Area 2 is a zoom of the initial and final position.

In Table 6.3 are represented the values of the evaluation made for the different methods in this scenario. Overall, the most consistent method is the MS(Odom+IPS) because its SD and RMSE values are within the lowest of all methods and the maximum and minimum mean values are within the highest. Methods using IPS show better evaluation results, as well as lower variations and errors, but it is important to emphasize that methods using IPS1 show better score than those using IPS2. Using the EKF produces a slight improvement in some results, but it is not very conclusive.

Method	Max(%)	Min(%)	SD	RMSE(%)
MS(Odom)	78.8632	78.6634	**0.0666**	21.2422
MS(EKF(Odom))	78.9131	78.7122	0.0752	21.1923
MS(EKF(Odom+IMU))	78.7841	78.4907	0.0811	21.3496
MS(Odom+IPS1)	**79.3313**	**79.1883**	**0.0467**	**20.7440**
MS(Odom+IPS2)	78.9774	**78.8445**	**0.0442**	21.0751
MS(EKF(Odom)+IPS1)	**79.5135**	**78.9510**	0.2015	**20.7398**
MS(EKF(Odom)+IPS2)	79.0796	78.6478	0.1260	21.0372
MS(EKF(Odom+IMU)+IPS)	**79.2592**	78.7921	0.1659	**20.9513**
MS(EKF(Odom+IMU)+IPS2)	78.9372	78.7219	0.0688	21.1609

Table 6.3: Table containing the evaluation data for the trajectories given by the different MSPF^2 methods. This data is relative to the mean of the points evaluation (with M1) of each trajectory. The numbers in bold represent the three best values in each column.

The methods with IMU are actually worse than the ones without, however this could be due to the absence of EKF covariance matrices tuning. In general the results do not show significant variations between them, so the interpretations from this test are not conclusive relatively to the intervention of the EKF, the IMU or the IPS.

6.3.2 Scenario 2

Since mobile robots, which use laser-based localization approaches, usually have problems in corridors [4] (*e.g.*, lack of features), the following test's goal was to check the behavior of the methods in this kind of scenario. Another problem for mobile robots is the existence of dynamic obstacles, so this scenario also tested if the methods would be affected by them.

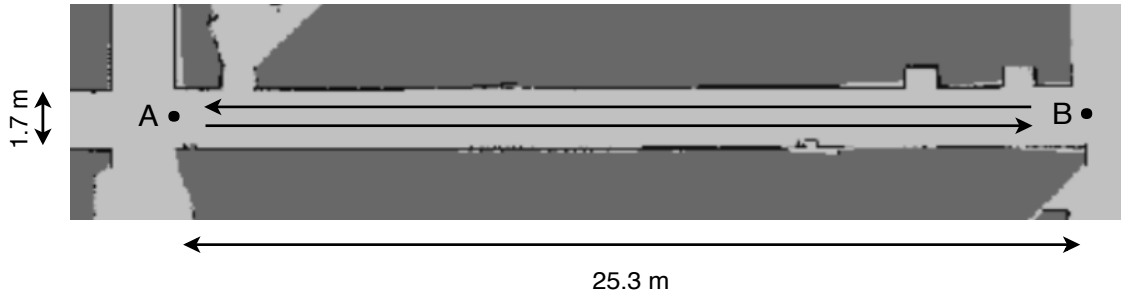


Figure 6.9: Part of the grid cell map used for this test, where A and B are respectively the initial and the final point of the corridor.

The test consisted on going from the corridor point A to B and from B to A, represented in Fig. 6.9, with different numbers of people passing in front of the robot from time to time. The number of people passing varied from one to five. Figure 6.10 represents the overlap of trajectories performed by all methods, ten times each.

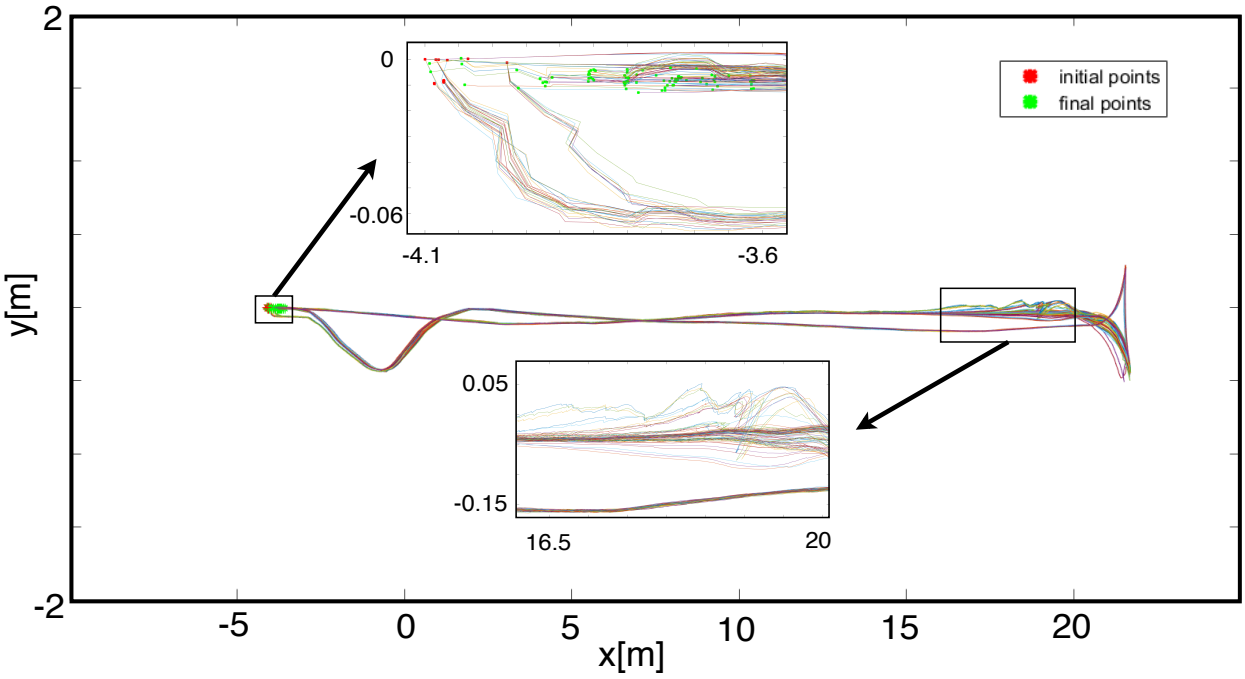


Figure 6.10: Representation of the overlapped trajectories from ten samples of all the MSPF^2 methods, with the red points being the initial points of each trajectory and the red ones, the last points.

As it can be seen in Fig. 6.10 the initial and final points of all trajectories are not too distant from each other, which means that the mobile robot finished the trajectories around the starting point, as intended. The variations are greater around the 20-meter zone, which could be originated by occlusions of the sensors caused by the people passing by or/and by the curved movements. Since this is not conclusive by itself, a further evaluation was conducted based on the score values that are represented in Table 6.4. The values of the evaluation made for the different methods in this scenario are represented in Table 6.4. The method that appears to be the most consistent is the MS(EKF(Odom)) because its SD and RMSE values are within the lowest of all methods and the maximum and minimum mean values are within the highest. The methods with IPS2 are within the best, but the ones with IPS1 are among the worst. This means that the methods with IPS1 are probably sampling particles in wrong places, due to sensor errors or occlusions, while the ones using IPS2 seem to maintain robustness in this more difficult scenario.

Method	**Max(%)**	**Min(%)**	**SD**	**RMSE(%)**
MS(Odom)	75.1633	**74.6857**	**0.1311**	24.5118
MS(EKF(Odom))	**75.3436**	**74.9133**	**0.1548**	**24.3975**
MS(EKF(OdomIMU))	74.7448	73.7588	0.2531	25.1183
MS(Odom+IPS1)	74.9945	73.9576	0.3328	24.5615
MS(Odom+IPS2)	**75.4011**	74.6368	0.2146	**24.4116**
MS(EKF(Odom)+IPS1)	75.3273	74.0383	0.3411	24.5158
MS(EKF(Odom)+IPS2)	**75.4217**	**74.8105**	**0.1961**	**24.4002**
MS(EKF(OdomIMU)+IPS1)	73.7810	71.2960	0.6410	26.0510
MS(EKF(OdomIMU)+IPS2)	74.6809	72.6072	0.4701	25.4912

Table 6.4: Table containing the evaluation data for the trajectories given the different MSPF^2 methods on the corridor. This data is relative to the mean of the points evaluation (with M1) of each trajectory. The numbers in bold represent the three best values in each column.

Using the EKF to just filter the odometry input makes a slight improvement in some results, which is consistent with the previous scenario. In general the methods using IMU seem to have worst score results and greater variation than the rest, which could be caused by a noisy contribution from the sensor or, as was mentioned, by the absence of EKF tuning.

In general these results still did not show significant variations between them, but they presented some differences relatively to the previous scenario. The results using IMU seem to be more conclusive in this scenario, because the sensor input seems to worsen the results. Based on the mean of the scores, filtering the odometry input with the EKF appears to improve the performance, but as the EKF was used in this dissertation, it did not significantly improve the localization. In the presence of measurement errors, the methods using IPS2 are more reliable than the ones using IPS1, however, the methods using IPS1 seemed consistently better in an environment with only static obstacles.

6.4 Recovery tests

The last test performed in this dissertation work was related to the method's capacity to recover from a localization loss (kidnapped robot problem). To test this, three of the previous used methods were chosen to test if the position can be recovered. The methods chosen for this test were the "MS(Odom)", the "MS(Odom+IPS1)" and the "MS(Odom+IPS2)".

These methods were chosen to verify the differences between having an IPS or not and to compare the two IPS methods, therefore, methods using EKF and/or IMU were not needed. The test consisted in giving the mobile robot a false initial position (in the x-axis) and observing if the estimated pose moved in the direction of the true initial position over time, with the chosen time interval being from 0[s] to 2[s]. The results are visualized in a ROS tool and since these results are snapshots, the time only stars counting after the first pose estimation, so that the first frame can contain a robot pose. The robot's poses are represented by red arrows, and the particles by blue points on the occupancy grid map.

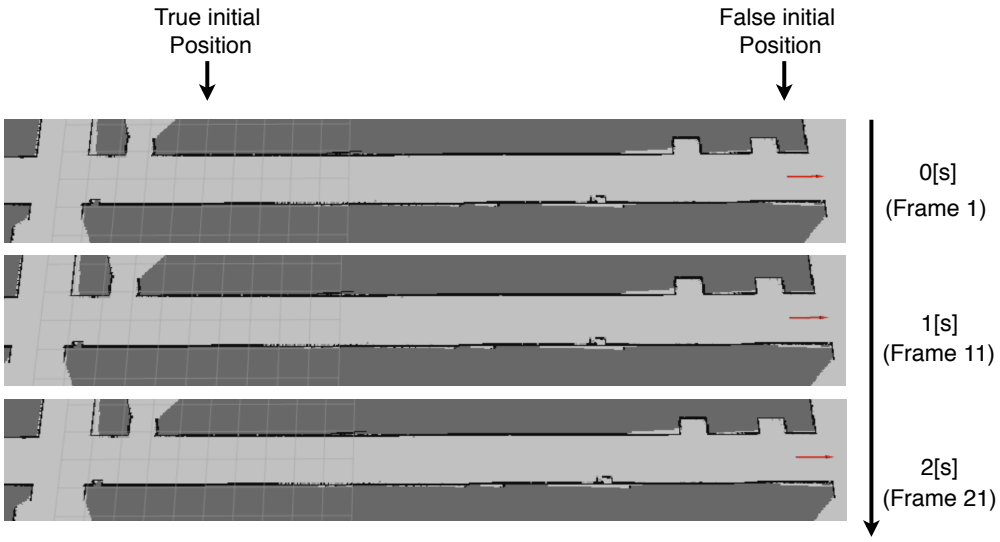


Figure 6.11: Recovery test performed with the "MS(Odom)" method. The snapshots are taken from the ROS rviz.

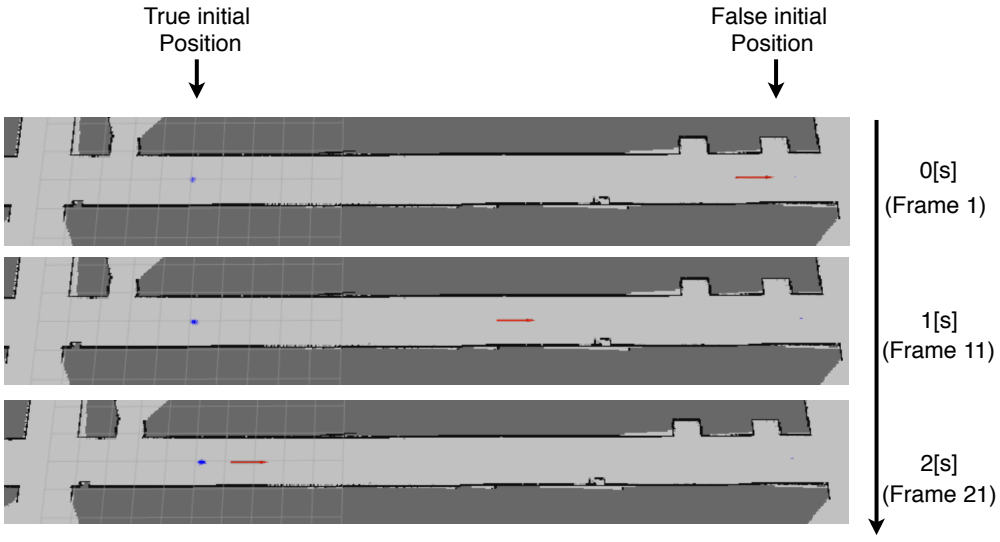


Figure 6.12: Recovery test performed with the "MS(Odom+IPS1)" method. The snapshots are taken from the ROS rviz.

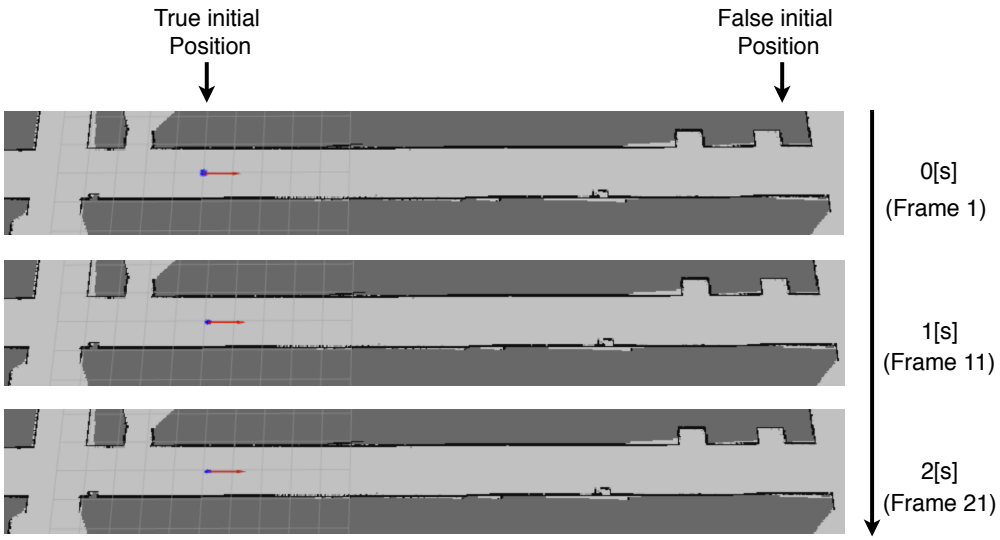


Figure 6.13: Recovery test performed with the "MS(Odom+IPS2)" method. The snapshots are taken from the ROS rviz.

Figures 6.11 to 6.12 show the recovery tests (represented on the occupancy grid map) performed by the three chosen methods. As it can be observed in Fig. 6.11, using only odometry data in the prediction stage, is not enough to recover from a localization loss, which is expected because this method moves the particles based only on motion estimation and an additional error. Figure 6.12 shows that using an IPS1 method moves the particles in the true position direction, but it is not ideal because only a percentage of the particles are created around the IPS measurements, which makes this recovery slightly slow. Using an IPS2 method, seems to be a better approach, since the particle positions are corrected at 0 seconds, which makes the recovery instantaneous.

Chapter 7

Conclusion and future work

7.1 Conclusion

The focus of this dissertation was the research of sensor fusion applied to PF algorithms for mobile robots (in this case, an AGV). For this purpose it was important to study different sensors, as well as different approaches on how to use these sensors individually or in a combined way. It was also necessary to test the mobile robot in both controlled and non-controlled environments to observe the differences in results. The software package was developed so it could be used in ROS and it does not rely on external software to obtain the location of the mobile robot. This package presents a robust behavior regardless of the different scenarios, since all of the trajectories given by the software corresponded to the intended paths. Even if the methods' estimation error was not tested, it is clear that it varies, depending on the used method.

Regarding the usage of EKF, the results appear to be better in some cases, but worse in others, so further testing is needed to understand if the EKF is advantageous for this software package. As for the IMU sensor, it does not present many advantages, which could be resolved by calibrating the EKF matrices, but, based on the current filter state, this sensor could be removed from the setup without interfering with the robustness of the software. Concerning the IPS, the results showed that the methods using IPS1 produced better results than those using IPS2, in scenarios with static obstacles, however, when dynamic obstacles were present, their performances were inverted. Nonetheless, methods using IPS presented scores among the best results of each tested scenario and these methods can solve localization loss problems, being that methods using IPS2 recover the pose faster that the ones using IPS1.

7.2 Future work

To improve the performance of this software package, there are some topics that could be further researched, such as:

- **EKF tuning**: Since the EKF package parameters were not tuned, this could be improved for better filtering of the sensor data.
- **Particle filter optimization**: The maximum time it takes for a iteration of the PF is $\simeq 100ms$. The algorithm could be optimized to have a faster response.
- **Map update**: The actual state of the PF algorithm, can only obtain the location of the mobile robot. An interesting idea would be to use the data from the PF to update the prior map in case of something is altered in the environment (*e.g.* boxes changing places).
- **Resampling**: Other methods could be tested to improve of the resampling step, so that the new set of particles, draws better samples.
- **Combination of the IPS methods**: As it was observed, both approaches of IPS fusion methods showed advantages and disadvantages but in different scenarios, so, the next step could be the fusion of both methods in order to improve the robustness of the proposed MSPF^2 package for all scenarios.

Bibliography

[1] H David. Why are there still so many jobs? the history and future of workplace automation. *Journal of Economic Perspectives*, 29(3):3–30, 2015.

[2] Alonso Kelly, Bryan Nagy, David Stager, and Ranjith Unnikrishnan. Field and service applications-an infrastructure-free automated guided vehicle based on computer vision-an effort to make an industrial robot vehicle that can operate without supporting infrastructure. *IEEE Robotics & Automation Magazine*, 14(3):24–34, 2007.

[3] Davide Ronzoni, Roberto Olmi, Cristian Secchi, and Cesare Fantuzzi. Agv global localization using indistinguishable artificial landmarks. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 287–292. IEEE, 2011.

[4] Sebastian Thrun, Dieter Fox, Wolfram Burgard, and Frank Dellaert. Robust monte carlo localization for mobile robots. *Artificial intelligence*, 128(1-2):99–141, 2001.

[5] Dieter Fox, Jeffrey Hightower, Lin Liao, Dirk Schulz, and Gaetano Borriello. Bayesian filtering for location estimation. *IEEE pervasive computing*, (3):24–33, 2003.

[6] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P. . Nordlund. Particle filters for positioning, navigation, and tracking. *IEEE Transactions on Signal Processing*, 50(2):425–437, Feb 2002.

[7] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[8] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[9] Ronald C Arkin and Robin R Murphy. Autonomous navigation in a manufacturing environment. *IEEE Transactions on Robotics and Automation*, 6(4):445–454, 1990.

[10] Humberto Martinez-Barbera and David Herrero-Perez. Development of a flexible agv for flexible manufacturing systems. *Industrial robot: An international journal*, 37(5):459–468, 2010.

[11] Humberto Martínez-Barberá and David Herrero-Pérez. Autonomous navigation of an automated guided vehicle in industrial environments. *Robotics and Computer-Integrated Manufacturing*, 26(4):296–311, 2010.

[12] R. G. Yudanto and F. Petré. Sensor fusion for indoor navigation and tracking of automated guided vehicles. In *2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–8, Oct 2015.

[13] C. Cosma, M. Confente, M. Governo, and R. Fiorini. An autonomous robot for indoor light logistics. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 3003–3008 vol.3, Sept 2004.

[14] Johann Borenstein, Hobart R Everett, Liqiang Feng, and David Wehe. Mobile robot positioning: Sensors and techniques. *Journal of robotic systems*, 14(4):231–249, 1997.

[15] Sok-Yong Lee and Hai-Won Yang. Navigation of automated guided vehicles using magnet spot guidance method. *Robotics and Computer-Integrated Manufacturing*, 28(3):425–436, 2012.

[16] M. E. Rusli, M. Ali, N. Jamil, and M. M. Din. An improved indoor positioning algorithm based on rssi-trilateration technique for internet of things (iot). In *2016 International Conference on Computer and Communication Engineering (ICCCE)*, pages 72–77, July 2016.

[17] Pablo Cotera, Miguel Velazquez, David Cruz, Luis Medina, and Manuel Bandala. Indoor robot positioning using an enhanced trilateration algorithm. *International Journal of Advanced Robotic Systems*, 13(3):110, 2016.

[18] Rainer Mautz. Indoor positioning technologies. 2012.

[19] Lei Zhu, Sheng Sun, and W. Menzel. Ultra-wideband (uwb) bandpass filters using multiple-mode resonator. *IEEE Microwave and Wireless Components Letters*, 15(11):796–798, Nov 2005.

[20] J. Lee, Y. Su, and C. Shen. A comparative study of wireless protocols: Bluetooth, uwb, zigbee, and wi-fi. In *IECON 2007 - 33rd Annual Conference of the IEEE Industrial Electronics Society*, pages 46–51, Nov 2007.

[21] E. Aitenbichler and M. Muhlhauser. An ir local positioning system for smart items and devices. In *23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings.*, pages 334–339, May 2003.

[22] Ramon F Brena, Juan Pablo García-Vázquez, Carlos E Galván-Tejada, David Muñoz-Rodriguez, Cesar Vargas-Rosales, and James Fangmeyer. Evolution of indoor positioning technologies: A survey. *Journal of Sensors*, 2017, 2017.

[23] Yi-Chao Chen, Ji-Rung Chiang, Hao-hua Chu, Polly Huang, and Arvin Wen Tsui. Sensor-assisted wi-fi indoor location system for adapting to environmental dynamics. In *Proceedings of the 8th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 118–125. ACM, 2005.

[24] Mike Hazas and Andy Hopper. Broadband ultrasonic location systems for improved indoor positioning. *IEEE Transactions on mobile Computing*, (5):536–547, 2006.

[25] Mike Hazas and Andy Hopper. Broadband ultrasonic location systems for improved indoor positioning. *IEEE Transactions on mobile Computing*, (5):536–547, 2006.

[26] Jeffrey K. Uhlmann Simon J. Julier. New extension of the kalman filter to nonlinear systems, 1997.

[27] Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter. *Kalman filtering and neural networks*, pages 221–280, 2001.

[28] N. Bellotto and H. Hu. Multisensor-based human detection and tracking for mobile service robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(1):167–181, Feb 2009.

[29] Sebastian Thrun, Dieter Fox, Wolfram Burgard, and Frank Dellaert. Monte carlo localization for mobile robots. In *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1999.

[30] Dieter Fox. Adapting the sample size in particle filters through KLD-sampling. *The International Journal of Robotics Research*, 22(12):985–1003, 2003.

[31] L. Zhang, R. Zapata, and P. Lépinay. Self-adaptive monte carlo localization for mobile robots using range sensors. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1541–1546, Oct 2009.

[32] Roland Siegwart, Kai O. Arras, Samir Bouabdallah, Daniel Burnier, Gilles Froidevaux, Xavier Greppin, Björn Jensen, Antoine Lorotte, Laetitia Mayor, Mathieu Meisser, Roland Philippsen, Ralph Piguet, Guy Ramel, Gregoire Terrien, and Nicola Tomatis. Robox at expo.02: A large-scale installation of personal robots. *Robotics and Autonomous Systems*, 42(3):203 – 222, 2003. Socially Interactive Robots.

[33] Gunhee Kim, Woojin Chung, Kyung-Rock Kim, Munsang Kim, Sangmok Han, and R. H. Shinn. The autonomous tour-guide robot jinny. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 4, pages 3450–3455 vol.4, Sept 2004.

[34] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige. The office marathon: Robust navigation in an indoor office environment. In *2010 IEEE International Conference on Robotics and Automation*, pages 300–307, May 2010.

[35] Ran Liu, Chau Yuen, Tri-Nhut Do, Dewei Jiao, Xiang Liu, and U-Xuan Tan. Cooperative relative positioning of mobile users by fusing IMU inertial and UWB ranging information. *CoRR*, abs/1704.01397, 2017.

[36] R. Liu, C. Yuen, T. N. Do, W. Guo, X. Liu, and U. X. Tan. Relative positioning by fusing signal strength and range information in a probabilistic framework. In *2016 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6, Dec 2016.

[37] Cyrill Stachniss and Wolfram Burgard. Particle filters for robot navigation. *Foundations and Trends® in Robotics*, 3(4):211–282, 2014.

[38] Gregory Dudek and Michael Jenkin. *Computational principles of mobile robotics*. Cambridge university press, 2010.

[39] D. U. Lee, J. D. Villasenor, W. Luk, and P. H. W. Leong. A hardware gaussian noise generator using the box-muller method and its error analysis. *IEEE Transactions on Computers*, 55(6):659–671, June 2006.

[40] J. Rodrigues Dias. A simple generalization of the box–muller method for obtaining a

pair of correlated standard normal variables. *Journal of Statistical Computation and Simulation*, 80(9):953–958, 2010.

[41] J. E. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25–30, 1965.

[42] R. Douc and O. Cappe. Comparison of resampling schemes for particle filtering. In *ISPA 2005. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005.*, pages 64–69, Sept 2005.

[43] T. Li, M. Bolic, and P. M. Djuric. Resampling methods for particle filtering: Classification, implementation, and strategies. *IEEE Signal Processing Magazine*, 32(3):70–86, May 2015.

[44] J. D. Hol, T. B. Schon, and F. Gustafsson. On resampling algorithms for particle filters. In *2006 IEEE Nonlinear Statistical Signal Processing Workshop*, pages 79–82, Sept 2006.

[45] Thomas Moore and Daniel Stouch. A generalized extended kalman filter implementation for the robot operating system. In Emanuele Menegatti, Nathan Michael, Karsten Berns, and Hiroaki Yamaguchi, editors, *Intelligent Autonomous Systems 13*, pages 335–348, Cham, 2016. Springer International Publishing.

[46] Sebastian Thrun and Arno Bücken. Integrating grid-based and topological maps for mobile robot navigation. In *Proceedings of the National Conference on Artificial Intelligence*, pages 944–951, 1996.

[47] Stefan Kohlbrecher, Johannes Meyer, Thorsten Graber, Karen Petersen, Uwe Klingauf, and Oskar von Stryk. Hector open source modules for autonomous mapping and navigation with rescue robots. pages 624–631. Springer, 2013.

[48] Nasser M. Nasrabadi. Pattern recognition and machine learning. *Journal of Electronic Imaging*, 16, 2007.

[49] Hokuyo. *Scanning Laser Range Finder UTM-30LX/LN Specification.* 2008.

[50] Xsens. *MTi User Manual.* 2014.

[51] Marvelmind Robotics. *Marvelmind Indoor Navigation System Operating manual.* 2018.

[52] Gaurav Pandey, James R McBride, and Ryan M Eustice. Ford campus vision and lidar data set. *The International Journal of Robotics Research*, 30(13):1543–1552, 2011.

Appendix A

Extra content

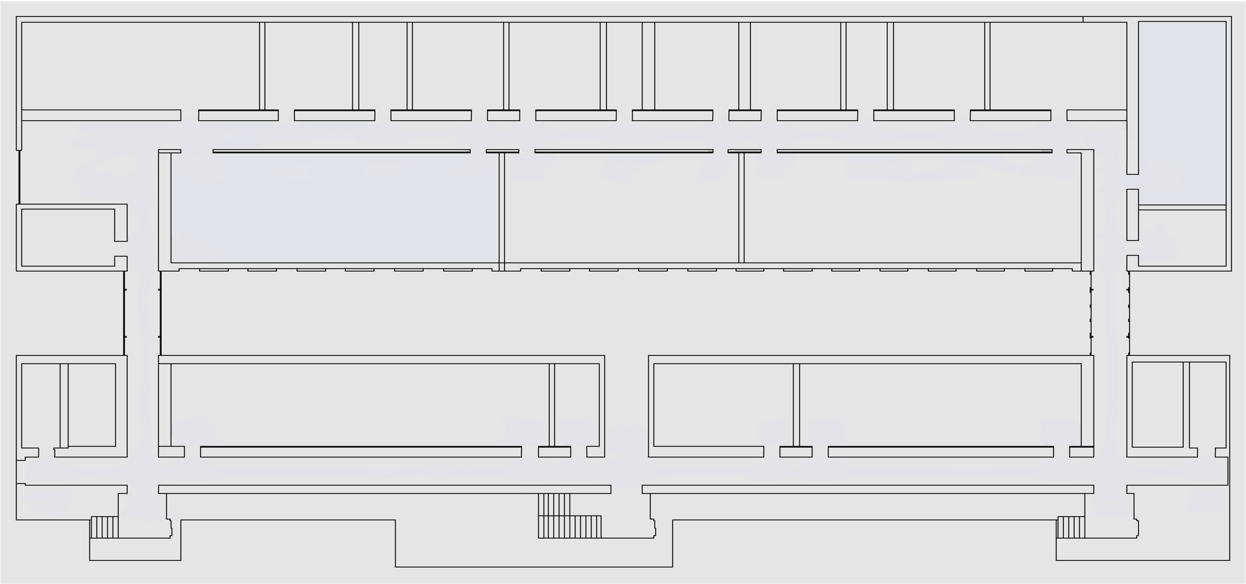


Figure A.1: Blueprint of the ISR-UC floor 0.

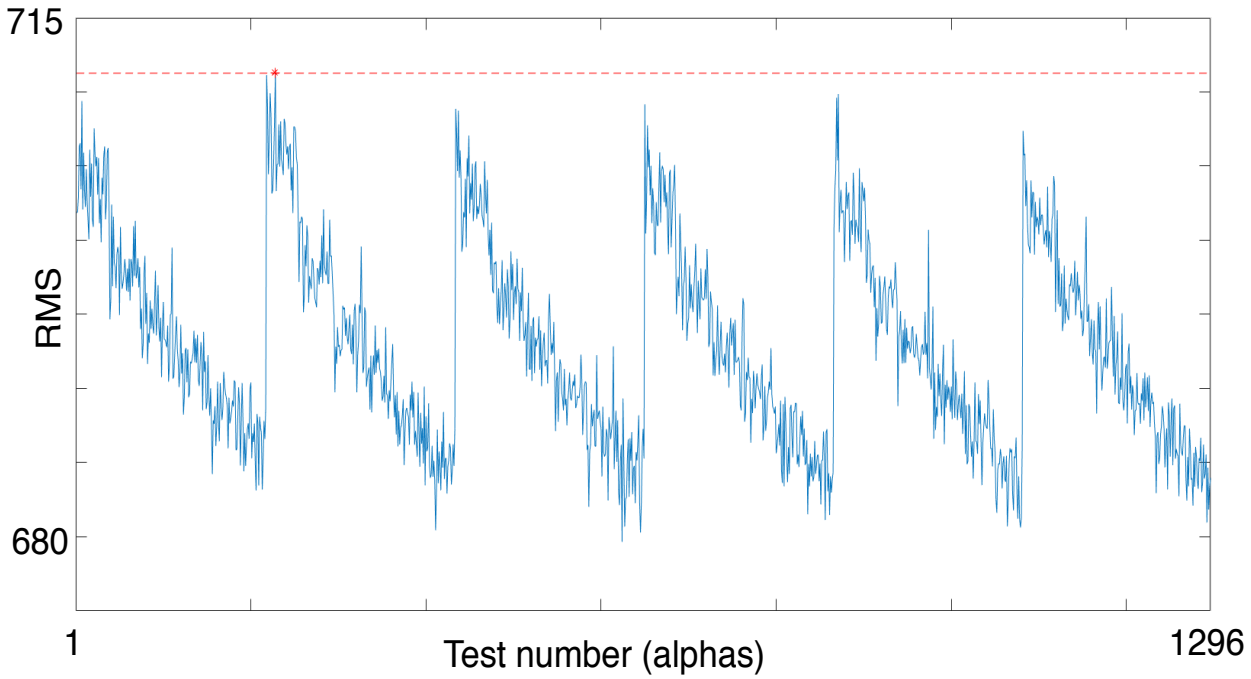


Figure A.2: Score's RMS of the tests performed for different values of the α_1, α_2, α_3 and α_4 parameters.

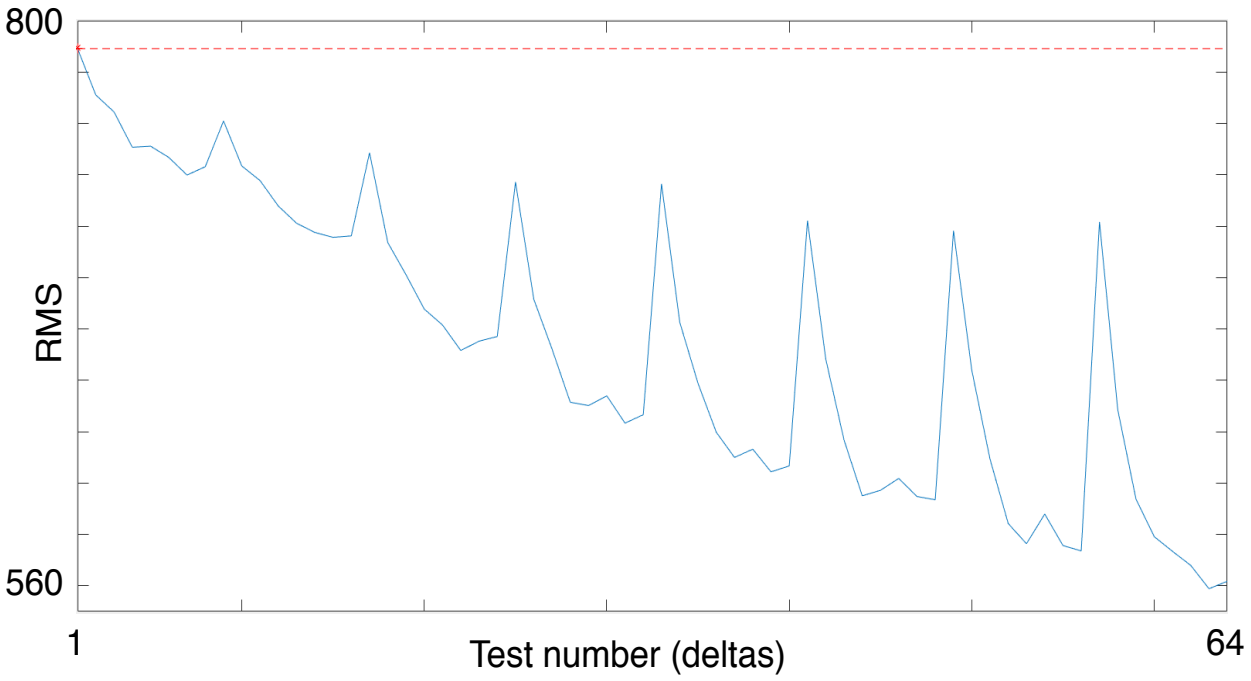


Figure A.3: Score's RMS of the tests performed for different values of the Δd and $\Delta\theta$ parameters.

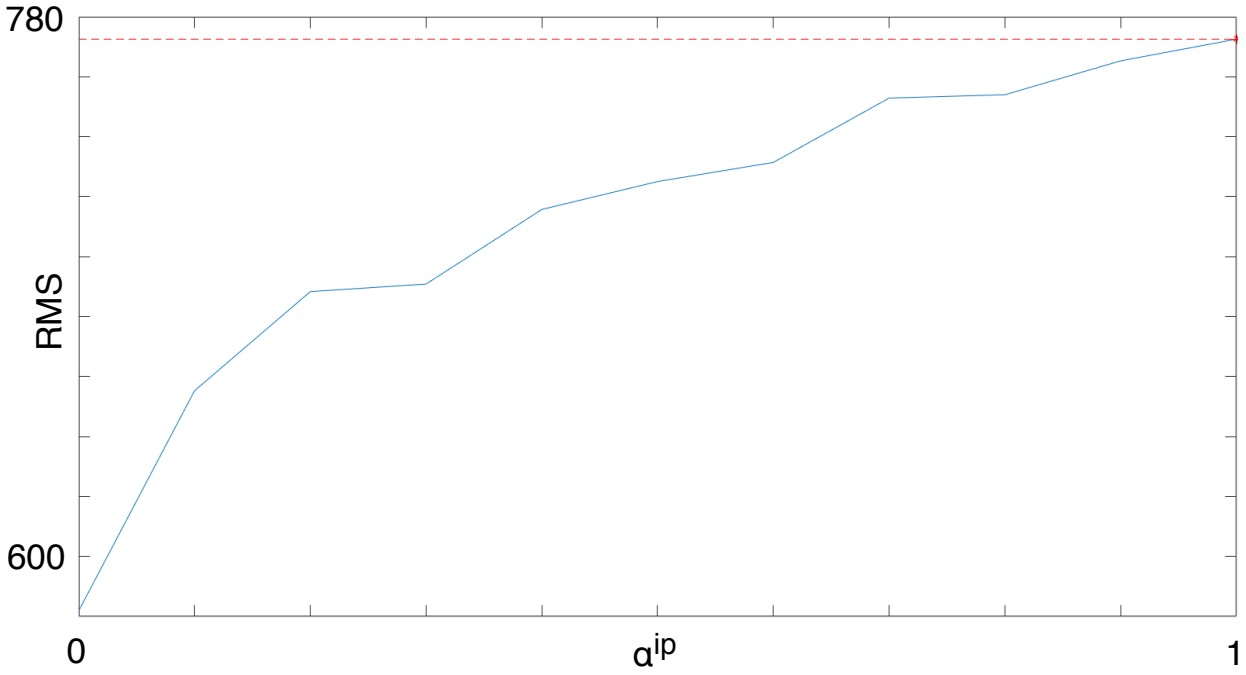


Figure A.4: Score's RMS of the tests performed for different values of the α^{ip} parameter.

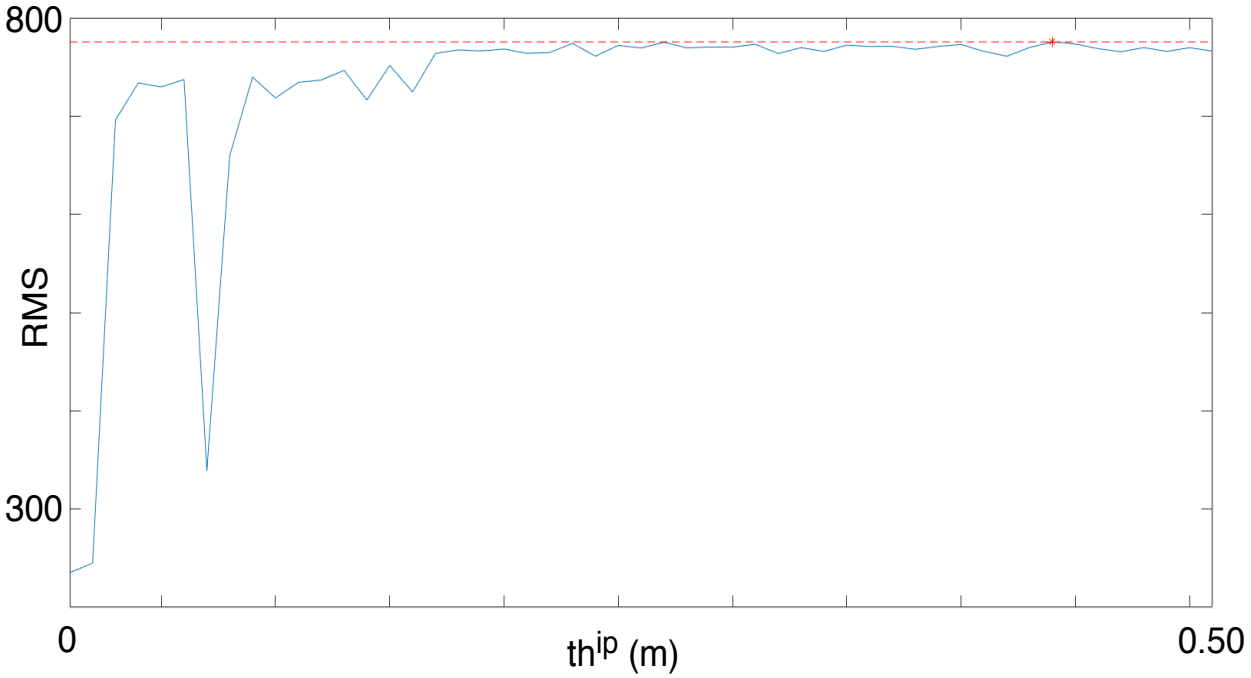


Figure A.5: Score's RMS of the tests performed for different values of the th^{ip} parameter.

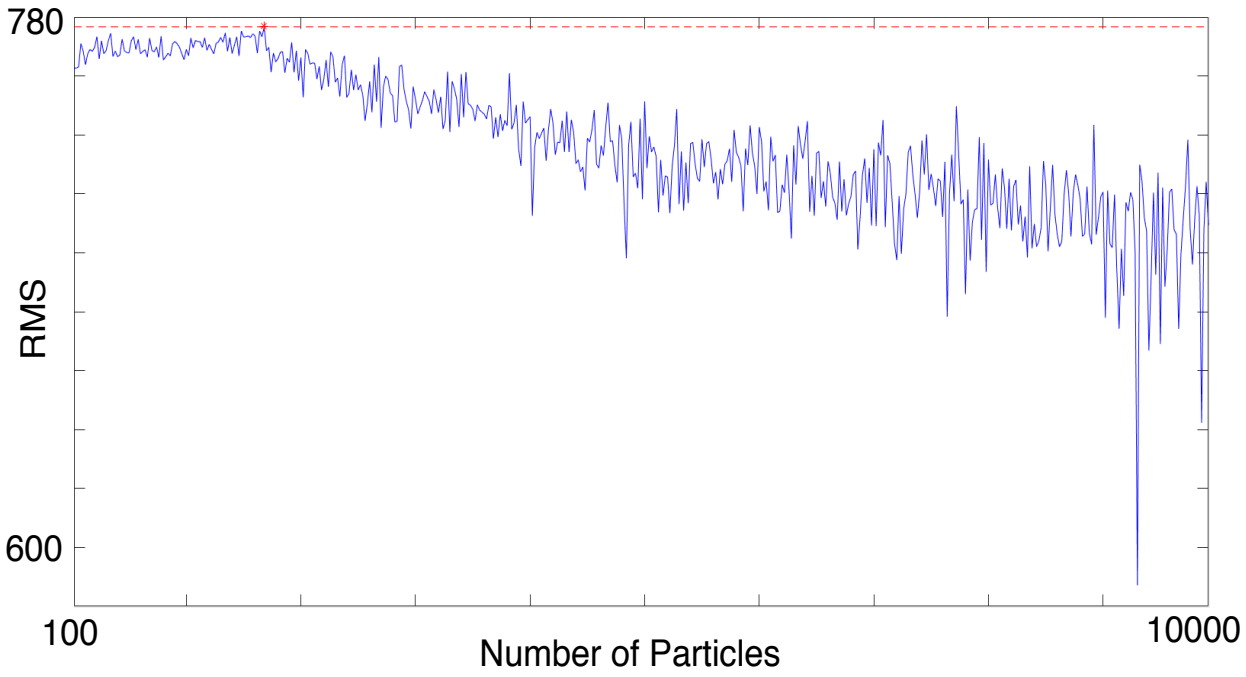


Figure A.6: Score's RMS of the tests performed for different values of the maximum number of particles parameter.