TECHNICAL UNIVERSITY OF CRETE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING
DIVISION OF TELECOMMUNICATIONS

Unmanned Vehicle Navigation in GPS-Denied Environments

by

Georgios Apostolakis

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DIPLOMA OF
ELECTRICAL AND COMPUTER ENGINEERING

July 2020

THESIS COMMITTEE

Professor Aggelos Bletsas, *Thesis Supervisor*
Associate Professor Michail G. Lagoudakis
Associate Professor Antonios Deligiannakis

Abstract

This work summarizes the most important localization and simultaneous localization and mapping (SLAM) algorithms for indoor navigation and studies a RFID-based technique for improved localization accuracy. Stereo cameras and LiDARs are widely used to assist a robot's movement inside a known or unknown environment, as they are not easily affected by the environmental noise. On the other hand, radio frequency identification (RFID) tags are very cheap and do not need any external power source, despite being affected a lot by the propagation environment. These characteristics have rendered them very popular and great research interest has risen in order to create accurate models for them. This thesis tests an approach which combines two measurements from each RFID tag, the received signal strength indication (RSSI) and the phase, in order to improve the localization/SLAM accuracy, compared to prior art. This integration is implemented by a particle filter, with an anchor tag being essential to compute the model parameters. For good estimation of the parameters, it is essential that the tags' measurements are correlated, i.e. tags' inter-distance should be less than half of the wavelength. Performance of the RSSI-phase integration was evaluated by comparing the estimation error to corresponding algorithms, which only use the RSSI or the phase. Localization error under light multipath (i.e. reflections of the radiation on surfaces of the area e.g. walls are combined and create a different than the expected signal to be received by the reader) was found in the order of less than 20 cm for all three algorithms, inside a 3D area of 12 m^3 volume. However, the error of the algorithm which only used the phase, remained significantly higher (more than 40 cm) for some time after its beginning. Localization error under strong multipath was found in the order of 20 cm when only phase was used, 60 cm when only RSSI was used and 40 cm with their combination, inside a 3D area of the same volume with above. Again, the phase algorithm induced a great error (more than 1.5 m) before it converged. Therefore, the proposed approach offers a reduced localization error from the initial time steps and can be implemented in environments with both light and rich multipath.

Acknowledgements

Firstly, I would like to thank my supervisor, Professor Aggelos Bletsas for his valuable guidance throughout all this work.

Also, Maria M. for all the encouragement and support through the last year, without which this work would have been much more difficult.

My friends and colleagues from the lab, especially Vaggelis G., Manos A., Iosif V., Roza Ch., George V., Kostas S., Vaggelis K. who helped me very much with their ideas, suggestions and technical issues.

My family and my close friends for all the support and help that they offered.

Contents

Chapter 1

Introduction

1.1 The indoor navigation problem

Navigation is the set of techniques required so that a robot can travel between target points inside an environment. It can be divided into 3 sub-problems[1]:

- **Localization of the robot** inside the (known or unknown) environment. This process consists of the computation of the robot's coordinates relatively to the environment's (global) coordinate system. When the environment is unknown, Simultaneous Localization And Mapping (SLAM) has to be implemented instead of localization (i.e. apart from detecting its position, the robot also constructs a map of its environment). It is essential that the robot finds a way to perceive some characteristics of its surroundings, so that it can relate itself with them (achieved by the use of sensors) or an external observer to exist and track the robot's position.

- **Decision about a target location to go.** This problem depends on multiple factors as the kind of the robot, the area where it moves, the purpose of the movement etc. There are numerous solutions, as numerous problems exist.

- **Path planning and transportation from its current to the desired location.** There is a great variety of path planning algorithms[2], with each one having advantages and disadvantages, depending on the case that it will be used. Motion control of the robot happens by giving the appropriate instructions to its actuators (e.g. wheels, arms).

This work assumes that a complete positioning system (such as GPS) is not available, thus the robot has to decide its position by observing the environment and measuring its own motion. Those measurements are processed by an inference algorithm (e.g. Kalman or particle filters) and finally an estimation for the current state is made. Some of the most effective ways to achieve that in indoor environments is a stereo camera or a LiDAR sensor (which can detect the distance from the surrounding objects) or RFID tags (which are also very cheap and can be placed in multiple anchor points of the area) along with an RFID reader. This thesis focuses on the first subproblem of the navigation procedure (localization & mapping) and makes an effort to improve the existing algorithms that use RFID observations.

1.2 Navigation using a stereo camera or a LiDAR sensor

A stereo camera (see Section 3.2) and a Light Detection And Ranging (LiDAR) sensor (see Section 3.1) use different technologies, but all measure the visual distance from an area's objects. These measurements are very accurate and can be easily used in localization or SLAM algorithms. For example, the Robot Operating System (ROS) framework (see Section 3.3) has many ready-to-use packages with such algorithms implemented, and they can be easily combined with many commercial sensors. From the view of the system's cost, the more expensive an option is, the more accurate and long-range observations it produces. Generally, stereo cameras are cheaper than the LiDAR sensors and can be even constructed by using two common monocular cameras and the appropriate algorithms. The disadvantage of the cameras is their scanning area, which is more constrained since the LiDARs usually cover a much wider angle (almost 360°) of the 2D space.

1.3 Navigation using RFID tags

The use of the RFID technology (see Section 4.1) is getting more and more popular nowadays. Thanks to the extremely low price of the RFID tags they can be massively used in large storage rooms (libraries, markets, etc.) to identify the positions of the objects, as well as some anchor points of the area. Many tasks previously done by humans can now be automated, since the robots can accurately know their location and where each object is. That's the reason for the continuing research to improve the existing algorithms and increase the accuracy as much as possible. Especially when combined with other sensors (e.g. camera), the RFID technology is a key to a robust, efficient indoor navigation solutions [3] [4] [5] [6] [7].

1.4 Navigation using other sensors

Apart from stereo cameras, LiDARs and RFIDs, a great variety of other sensors can be used. The most common are:

- **Infrared (IR) sensors:** They measure distances by using triangulation (detection of the signal's arrival angle and computation the distance from the object to which it was reflected). Their range and accuracy is worse than those of the LiDAR sensors.
- **Ultrasonic sensors:** They measure distances by using sound waves (counting of the time since the wave was emitted until it returns back). They are less accurate than the IR sensors, and with even smaller range.
- **Barometers:** They can be used to compute the height of a robot from the ground. However, when inside the room air waves exist (e.g. drone flying,

open doors/windows), barometers are affected too much and their accuracy is reduced.

- **Compass:** Computes the orientation of the robot, relatively to the magnetic field of the earth. However, it is useless in places with strong magnetic fields.
- **Monocular cameras:** They can be used to detect landmarks of the area and, in combination with the robot's motion, to compute its current pose.
- **Radio Frequency (RF) Beacons:** They are small devices emitting RF pulses and they can be considered as "an indoor GPS system" since they emit signals (just as the GPS satellites) which are detected and processed by the robot. The only difference is the way of processing, due to the different distances between the transmitter and the receiver in each case. Furthermore, they require a power source making their installation more complex (and expensive) than RFID tags.
- **Global Positioning System (GPS) sensors:** For outdoor places without many obstacles, GPS sensors offer accurate localization without the need of transmitting radiation.

1.5 Thesis outline

This work summarizes some of the main indoor localization and SLAM techniques, as well as investigates the possibilities of RFID-based algorithms. Chapter 2 analyzes both the localization and the SLAM problem, as well as the most famous algorithms to solve them. In Chapter 3 the technology of stereo vision and LiDAR sensors is explained thoroughly, as well as the ROS framework is summarized, through which it is possible to easily use pre-implemented algorithms with a great variety of sensors. The technology of RFIDs, some significant papers on their use for navigation and a new approach for accuracy increase are discussed thoroughly in Chapter 4. Chapter 5 quotes some maps produced by the 'Google Cartographer' algorithm in ROS, as well as the efficiency of the new approach proposed in Chapter 4. Finally, Chapter 6 gives some conclusions and possible directions for future work.

Chapter 2

Navigation algorithms

2.1 The localization problem

Localization is the problem of estimating a robot's pose (i.e. position and orientation) inside a known environment. The estimation is based on measurements taken by internal or external robot sensors which are analyzed using inference methods. Dividing the localization problems into categories can be achieved by using many different criteria, such as[8]:

- **The type of knowledge available initially and at run-time**:
  - *Position tracking*: The initial pose of the robot is known and the robot is not static. Its path is tracked using its own motion (i.e. the instructions that rule its movement inside the environment) or/and observations of the robot relatively to anchor points in the environment. The greater the noise is, the greater the uncertainty of the final estimation is.
  - *Global localization*: The same as the position tracking problem, with the difference that the initial pose of the robot is unknown. A different than the above approach has to be used to estimate the robot's pose, by using anchor points of the environment (and the robot's motion too, if the robot is moving). This is a harder problem to solve since the available knowledge is limited.
  - *Kidnapped robot problem*: The same as the global localization problem with the assumption that anytime, the robot may be kidnapped by an external force and moved to another location. That makes the problem even more difficult than global localization. On the one hand this is not a situation that often occurs in robotics, but on the other hand, the solution to this problem makes an algorithm capable of recovering from failures (something that is not guaranteed by the state-of-the-art localization algorithms).
- **The type of environment around the robot**:
  - *Static environments*: The only changing thing in such environments is the robot when it moves.
  - *Dynamic environments*: While time passes parts of the environment are changing, rapidly or slowly (e.g. people walking, objects moved from one location to another).

- **The use of the robot's motion**:
  - *Active localization*: The algorithm can control the robot and drive it at any desired location and perform locomotion actions.
  - *Passive localization*: The algorithm is not capable of controlling the robot.
- **The number of robots to localize**:
  - *Single-robot localization*: All data being collected concerns the same robot which needs to be localized.
  - *Multi-robot localization*: This type of localization refers to teams of robots which are able to detect and communicate with each other. Then, their belief for each other can furthermore improve the single-robot estimation computed from each robot for itself.

2.1.1 EKF Localization

EKF localization[8] uses an Extended Kalman Filter to fuse the measurements of the robot's motion (e.g. its IMU) with the observations of landmarks in the environment. This filter uses a Gaussian approximation of the real model, i.e. a mean and a covariance matrix.

The algorithm (see Algorithm 1) consists of two discrete steps:

1. **Prediction step:** The motion of the robot is used to change the previous estimation (mean and covariance matrix), depending on its movement at that time. This motion contains a significant amount of noise and increases the variance of the estimation.
2. **Correction step:** The observations of the environment are used to 'correct' the prediction of the previous step (i.e. slightly change the mean and decrease the variance of the estimation).

In the case which the landmarks cannot be distinguished from each other, a way for association of the observations with the known map has to be found. Many techniques exist for that reason, such as Multi-Hypothesis Tracking. This strategy uses multiple Gaussians, with each one expressing a different association between the observations and the map of the landmarks. The final estimation is the weighted average of those Gaussians or the one with the maximum weight.

Another issue interesting to mention is the initialization of the algorithm's parameters. This is not too important, since the algorithm will converge to the right estimation anyway. However, the more accurate the initial guess is, the faster the algorithm will converge.

Algorithm 1 Extended Kalman Filter (EKF) Localization

Required:

μ_{t-1}, the estimated mean of the previous time step
Σ_{t-1}, the covariance matrix of the previous time step
u_t, the robot's motion (odometry measurement) for the current time step
z_t, the observations of the environment's landmarks for the current time step
m, the set of known landmarks in the environment

Returned:

μ_t, the estimated mean of the current time step
Σ_t, the covariance matrix of the current time step

Pseudocode:

1: $\hat{\mu}_t \leftarrow g_t(u_t, \mu_{t-1})$ ▷ Prediction step
2: $\hat{\Sigma}_t \leftarrow G_t \Sigma_{t-1} G_t^T + R_t$
3: **for** $j \leftarrow 1 : size\,(z_t)$ **do** ▷ Correction by using all the observations
4: find ℓ such that z_t^j refers to map landmark m_ℓ
5: $\hat{z}_t^j = h_t^j\,(\hat{\mu}_t, m_\ell)$
6: $K_t^j \leftarrow \hat{\Sigma}_t \left(H_t^j\right)^T \left(H_t^j \hat{\Sigma}_t \left(H_t^j\right)^T + Q_t\right)^{-1}$ ▷ Kalman Gain
7: $\hat{\mu}_t \leftarrow \hat{\mu}_t + K_t^j \left(z_t^j - \hat{z}_t^j\right), \quad \hat{\Sigma}_t \leftarrow \left(I - K_t^j H_t^j\right) \hat{\Sigma}_t$ ▷ Correction step
8: **end for**
9: $\mu_t \leftarrow \hat{\mu}_t, \quad \Sigma_t \leftarrow \hat{\Sigma}_t$

Explanation of the symbols:

- Function g_t is the transition model of the robot, i.e. changes its estimated location according to its motion u_t.
- Matrix G_t is the Jacobian of the function g : $G_t = \frac{\partial g_t(u_t, \mu_{t-1})}{\partial \mu_{t-1}}$.
- Matrix R_t is the noise covariance matrix of the robot's motion.
- Function h_t^j is the observation model of the robot for the map feature j, i.e. a non-linear function, for which it is true that: $\bar{z}_t^j = h_t^j(x_t, m_\ell)$. This function computes the expected value of the observation z_t^j based on the filter's estimation for the robot's pose x_t and the map feature m_ℓ to which the observation corresponds.
- H_t^j is the Jacobian matrix of the function h_t^j : $H_t^j = \frac{\partial h_t^j(\hat{\mu}_t, m_\ell)}{\partial \hat{\mu}_t}$.
- Matrix Q_t is the noise covariance matrix of the observations z_t^j.

Unscented Kalman Filter (UKF) Localization[8] is an improvement of EKF localization, where a UKF is used instead of an EKF. UKF does not just use a mean and a covariance matrix to represent each of its states, but also some additional components which increase the accuracy of the estimation. This accuracy increase can be justified by the better modeling of the impact of the noise coming from the various sensors, instead of a simple Gaussian approximation.

2.1.2 Grid Localization

Grid localization[8] uses a discrete approximation (histogram filter), in which the space of all possible states (e.g. locations/poses to be estimated) is divided into a grid of K cells. Each cell is processed separately from the rest, and that procedure is also comprised by two stages; the prediction and the correction one, similarly to EKF localization.

Algorithm 2 illustrates the localization procedure, with $f_{motion}(...)$ and $f_{landmark}(...)$ being the motion model and the observation model, respectively. Their computation depends on the sensors, the environment and the robot.

The resolution of the grid is a key factor to the accuracy of the localization. High resolution induces high accuracy (supposing that the observations are accurate enough too), but also induces a high computational cost due to the great number of the grid's cells. Lower grid resolution also reduces both of those metrics. Thus, the optimal trade-off has to be decided relatively to the available resources and the required accuracy.

Algorithm 2 Grid Localization

Required:

$p_{t-1}^{1,2,...,K}(x_{1,2,...,K})$, the posterior belief for each cell $x_1, x_2, ..., x_K$ at the previous time step

u_t, the robot's motion (odometry measurement) for the current time step

z_t, the observations of the environment's landmarks for the current time step

m, the set of known landmarks in the environment

Returned:

$p_t^{1,2,...,K}(x_{1,2,...,K})$, the posterior probability for each cell $x_1, x_2, ..., x_K$ at the current time step

Pseudocode:

1: **for** $k \leftarrow 1 : K$ **do**
2: $\quad \hat{p}_t^k(x_k) \leftarrow \sum_{kk=1}^{K} p_{t-1}^{kk}(x_{kk})\, f_{motion}(x_{kk}, x_k, u_t)$ ▷ Prediction step
3: $\quad p_t^k(x_k) \leftarrow \eta \hat{p}_t^k(x_k)\, f_{landmark}(m, x_k, z_t)$ ▷ Correction step
4: **end for**

Explanation of the symbols:

- Function $f_{motion}(x_{src}, x_{dst}, u_t)$ returns the probability of transitioning from cell x_{src} to cell x_{dst} with the robot's motion being equal to u_t.
- Function $f_{landmark}(m, x_k, z_t)$ returns the probability of getting the observation z_t from the cell x_k inside the map m.
- η is a normalization factor so that $\sum_{k=1}^{K} p_t^k(x_k) = 1$.

2.1.3 Monte Carlo Localization

Monte Carlo localization[8] is one of the most popular localization algorithms in robotics (see Algorithm 3). Based on the use of a particle filter it can approximate multimodal distributions, something very useful in situations where there are areas in the map very similar to each other. The algorithm works for static environments, solves the global localization problem and can be used with both active and passive approaches. An important parameter to be set is the number of the particles used by the filter (sample size). The more particles to be used, the more accurate the estimation to be. On the other hand, the less particles to be used, the more computationally-efficient the implementation to be. That trade-off defines a need for adaptation of the sample size to every occasion. What is more, there is a technique called Kullback-Leibler Divergence (KLD)[9], which automatically adapts the sample size in a way that the estimation error remains under a certain limit. That technique uses far less particles than the respective fixed-sample-size algorithm with the same accuracy.

Resampling is a very important step of the algorithm, because it changes the proposal distribution (particles) in order to approximate more the target distribution (real). Appendix B gives more details on that process, as well as some methods to improve its efficiency.

Solving the kidnapped robot problem (equivalently recovering from failures) in Monte Carlo localization requires just a slight modification. That is, at the resampling step to add some random particles to prevent particle deprivation, when the robot is kidnapped. The amount of the random particles may be fixed-size or adaptive by comparing the short-term with the long-term likelihood of the observations (Augmented MCL[8][10]).

Algorithm 3 Monte Carlo Localization (MCL)

Required:

X_{t-1}, the sample set of the previous time step
M, the size of the sample set
m, the map of the environment
u_t, the robot's motion (odometry measurement) for the current time step
z_t, the observations of the environment's landmarks for the current time step

Returned:

X_t, the new sample set
$\hat{X}_t$, position estimation for the current time step

Pseudocode:

1: $X_t = \tilde{X}_t = \{\}$
2: **for** $k \leftarrow 1 : M$ **do** ▷ Particle & Weight update
3: $x_t^{[k]} \sim motion\_model\_distribution\left(x_{t-1}^{[k]}, u_t\right)$
4: $w_t^{[k]} \leftarrow p\left(z_t | x_t^{[k]}, m\right)$ ▷ Measurement model
5: $\tilde{X}_t \leftarrow \tilde{X}_t + \{\langle x_t^{[k]}, w_t^{[k]} \rangle\}$
6: **end for**
7: **for** $k \leftarrow 1 : M$ **do** ▷ Resampling
8: choose i with probability $\propto w_t^{[i]}$ from $\tilde{X}_t$
9: $X_t \leftarrow X_t + \{x_t^{[i]}\}$
10: **end for**
11: $\hat{X}_t = \frac{1}{\sum_{k=1}^{M} w_t^{[k]}} \left(\sum_{k=1}^{M} w_t^{[k]} x_t^{[k]}\right)$ ▷ Position estimation

2.2 The SLAM problem

SLAM stands for Simultaneous Localization And Mapping [8]. The robot is placed inside an unknown environment and it has to create a map of it, as well as localize itself relatively to the map it built. It is obvious that in this problem far less data is available, and the number of the unknown variables is increased (compared to the localization problem), which makes it significantly more difficult.

Briefly, the SLAM algorithm can be described in 4 simple steps:

- Set the starting point of the robot's path as coordinate **0**.
- Use the robot's estimated pose to observe new landmarks on the environment and to add them to the map (with an uncertainty, due to the observation's error and the uncertainty on the robot's location).
- Observe the same landmarks multiple times to reduce the uncertainty on their location.
- Use the observation of the known landmarks (not observed for first time) to reduce the uncertainty on the robot's pose after each move.

The graphical model best describing the above procedure can be seen in figure 2.1, where:

- x_t denotes the robot's estimated pose at time t.
- u_t denotes the instructions which control the robot's movement (or equivalently the data from the robot's Inertial Measurement Unit) at time t.
- z_t denotes the set of landmark observations in the environment by the robot at time t.
- m denotes the robot's estimation for the map of the environment.

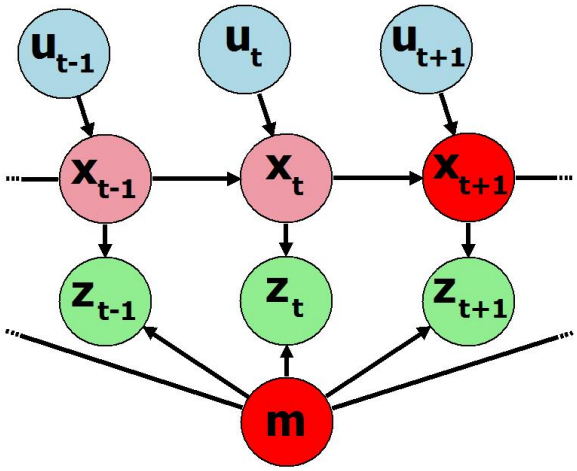


Figure 2.1: SLAM Graphical Model.

SLAM comes in 2 forms, accordingly to the amount of information that has to be estimated:

- **Online SLAM:** The algorithm returns the posterior probability over the current pose of the robot and the estimated map of the environment, i.e. $p\left(x_t, m | u_{1:t}, z_{1:t}\right)$.
- **Full SLAM:** The algorithm returns the posterior probability over the whole path of the robot as well as the estimated map of the environment, i.e. $p\left(x_{1:t}, m | u_{1:t}, z_{1:t}\right)$.

SLAM algorithms are computationally expensive. Especially the Full SLAM problem (which requires computation of the full posterior) is usually infeasible, due to the high dimensionality of the parameters' space and the large number of possible ways to match the (already observed) discrete landmarks with the measurements.

2.2.1 EKF SLAM

Extended Kalman Filter (EKF) SLAM[8] is historically the first SLAM algorithm developed and solves the online SLAM problem. The robot's motion as well as the

observations of the environment features are fused inside an EKF and an estimation of the map and the robot's pose is induced.

EKF uses a Gaussian approximation of the variables to be estimated. When those variables are close-to-linear, the EKF converges. However, in the real world there are cases in which linearity does not exist at all and at such cases the algorithm may diverge from the correct solution. What is more, EKF SLAM is generally computationally expensive, despite various techniques proposed to reduce its complexity. Thus, more efficient algorithms have been developed since then (e.g. FastSLAM) and EKF SLAM is not widely used anymore, but only in very specific cases.

The main procedure can be divided in two main stages, which are looped at each time step:

- **Prediction step:** The motion of the robot (e.g. the data coming from its IMU) is fused along with the estimation of the previous time step.
- **Correction step:** The observations of the environment's features are fused along with the estimation from the 'Prediction' step and improve the accuracy for the estimation of the map features and the robot's position.

Depending on the type of sensors, correspondence between observations and landmarks may or may not be a problem, with the correspondence likelihood (i.e. a function which returns how well a set of correspondences between observations and landmarks fits) to be computed when necessary and its global maxima to be used as the most probable set of correspondences. Algorithm 4 describes the EKF SLAM briefly. At each time step, it returns a vector with the means, and one covariance matrix, i.e. the Gaussian approximations of all the variables to be estimated:

$$\mu_t = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ m_1^1 \\ m_2^1 \\ \vdots \\ m_1^N \\ m_2^N \\ \vdots \end{bmatrix}, \qquad \Sigma_t = \begin{bmatrix} \Sigma_{xx} & \Sigma_{xm} \\ \Sigma_{mx} & \Sigma_{mm} \end{bmatrix} \tag{2.1}$$

x_i variables denote the coordinates of the robot, while m_i^k denote the i-th coordinate of the k-th map feature. Finally, Σ_{xm} denotes the sub-matrix with the covariances between each pair of x_i and m_i^k variable.

Relatively to the initialization of the filter, its state vector initially contains only the robot's coordinates, all equal to zero, and its covariance matrix is set to zero too.

Algorithm 4 Extended Kalman Filter (EKF) SLAM

Required:

μ_{t-1}, the vector with the estimation's means (pose + map) of the previous time step

Σ_{t-1}, the vector with the covariance matrix of the previous time step

u_t, the robot's motion (odometry measurement) for the current time step

z_t, the set of observations of map features for the current time step

Returned:

μ_t, the vector with the estimation's means (pose + map) of the current time step

Σ_t, the vector with the covariance matrix of the current time step

Pseudocode:

$\hat{\mu}_t \leftarrow \mu_{t-1} + F_x^T g_t(u_t, \mu_{t-1})$ ▷ Prediction step

$\hat{\Sigma}_t \leftarrow G_t \Sigma_{t-1} G_t^T + F_x R_t F_x^T$

for $j \leftarrow 1 : size\,(z_t)$ **do** ▷ Correction by using all the observations

    find m such that z_t^j refers to map landmark in position m of the state vector $\hat{\mu}_t$.

    if landmark m never seen before **then** ▷ Initialize new feature

        Initialize landmark m inside the filter's state and increase size of $\hat{\mu}_t, \hat{\Sigma}_t$

    end if

    $\hat{z}_t^j = h_t^j\,(\hat{\mu}_t, m)$

    $K_t^j \leftarrow \hat{\Sigma}_t \left(H_t^j\right)^T \left(H_t^j \hat{\Sigma}_t \left(H_t^j\right)^T + Q_t\right)^{-1}$ ▷ Kalman Gain

    $\hat{\mu}_t \leftarrow \hat{\mu}_t + K_t^j \left(z_t^j - \hat{z}_t^j\right), \quad \hat{\Sigma}_t \leftarrow \left(I - K_t^j H_t^j\right) \hat{\Sigma}_t$ ▷ Correction step

end for

$\mu_t \leftarrow \hat{\mu}_t, \quad \Sigma_t \leftarrow \hat{\Sigma}_t$

Explanation of the symbols:

- Matrix F_x maps the state vector of the robot to the state vector (robot's pose + map) of the filter.
- Function g_t is the motion model of the robot, i.e. changes its estimated location according to its motion u_t.
- Matrix G_t equals with: $G_t = I + F_x^T G_{0,t} F_x$, where $G_{0,t}$ is the Jacobian of the function g_t: $G_{0,t} = \frac{\partial g_t(u_t, \mu_{t-1})}{\partial \mu_{t-1}}$.
- Matrix R_t is the noise covariance matrix of the robot's motion.
- Function h_t^j is the observation model of the robot for the map feature j, i.e. a non-linear function, for which it is true that: $\bar{z}_t^j = h_t^j(\hat{\mu}_t, m)$. This function computes the expected value of the observation z_t^j based on the filter's estimation for the robot's pose x_t and the map feature m to which the observation corresponds. Both of them can be found in the state vector $\hat{\mu}_t$ which is an argument of the function.
- $H_t^j = H_{0,t}^j F_{x,j}$, where $H_{0,t}^j$ is the Jacobian matrix of the function h_t^j: $H_{0,t}^j = \frac{\partial h_t^j(\hat{\mu}_t, m)}{\partial \hat{\mu}_t}$, and $F_{x,j}$ maps $H_{0,t}^j$ to a higher-dimension space.
- Matrix Q_t is the noise covariance matrix of the observations z_t^j.

2.2.2 FastSLAM

FastSLAM [11] [8] is based on particle filters, and more specifically on their 'Rao-Blackwellized' version. The main assumption of this algorithm is that the map features are independent from each other, i.e. the following factorization of the full SLAM posterior is possible:

$$p\left(x_{1:T}, m | z_{1:T}, u_{1:T}\right) = p\left(m | x_{1:T}, z_{1:T}\right) p\left(x_{1:T} | z_{1:T}, u_{1:T}\right) =$$

$$= p\left(x_{1:T} | z_{1:T}, u_{1:T}\right) \prod_{i=1}^{N} p\left(m_i | x_{1:T}, z_{1:T}\right)$$

- $p\left(x_{1:T} | z_{1:T}, u_{1:T}\right)$ is estimated by the use of a particle filter and the distribution of the particles.
- $p\left(m_i | x_{1:T}, z_{1:T}\right), i = 1, 2, ..., N$ is estimated by the use of an EKF (per landmark) which fuses the new observations with the previous ones.

Note: x_t denotes the pose of the robot at time t, z_t the set of landmark observations at time t, u_t the robot's motion at time t and m_i the i-th map feature.

Online SLAM can also be solved with the change that, instead of $x_{1:T}$, each particle will track only the last estimation x_T for the robot's location.

The algorithm uses a particle filter to estimate both the pose of the robot and the location of the landmarks in the environment. Thus, each particle consists of the robot's coordinates, as well as the coordinates of all the detected map features (1 path and 1 map). Each feature is updated with the use of an Extended Kalman Filter, while the weight of each particle equals with the product of the individual weights resulting from the likelihood of the observations with the features. Correspondence of the landmarks to the observations may be a problem here too and, again, it can be solved by computing the correspondence likelihood (as explained in 2.2.1). Algorithm 5 demonstrates the basic procedure of FastSLAM.

Explanation of the symbols:

- Each particle p has the form $\langle x_t^{[p]}, \left(\mu_{1,t}^{[p]}, \Sigma_{1,t}^{[p]}\right), ..., \left(\mu_{N,t}^{[p]}, \Sigma_{N,t}^{[p]}\right)\rangle$ with $x_t^{[p]}$ being the robot's pose and $\left(\mu_{i,t}^{[p]}, \Sigma_{i,t}^{[p]}\right)$ being the estimation for the location of the map feature i, $i = 1, 2, ..., N$.
- Function h is is the observation model of the robot, i.e. a non-linear function, for which it is true that: $\bar{z}_t^j = h(x_t, \mu_m)$. This function computes the expected value of the observation z_t^j, based on the pose x_t of the robot and the mean μ_m of the estimation for a map feature.
- H is the Jacobian matrix of the function h: $H = \frac{\partial h(x_t, \mu_m)}{\partial \mu_m}$.
- Matrix Q_t is the noise covariance matrix of the observations z_t^j.

Algorithm 5 FastSLAM

Required:

Y_{t-1}, the set of particles from the previous time step

W_{t-1}, the vector with the particle weights from the previous time step

u_t, the robot's motion (odometry measurement) for the current time step

z_t, the set of observations of map features for the current time step

Returned:

Y_t, the set of particles from the current time step

W_t, the vector with the particle weights from the current time step

Pseudocode:

1: **for** $p \leftarrow 1 : size\,(Y_{t-1})$ **do** ▷ For all the particles

2: Retrieve $x_{t-1}^{[p]}$, $\left(\mu_{1,t-1}^{[p]}, \Sigma_{1,t-1}^{[p]}\right)$, ..., $\left(\mu_{N,t-1}^{[p]}, \Sigma_{N,t-1}^{[p]}\right)$ from particle $y^{[p]}$

3: $x_t^{[p]} \sim p\left(x_t | x_{t-1}^{[p]}, u_t\right)$ ▷ Prediction step

4: $w_t^{[p]} \leftarrow w_{t-1}^{[p]}$

5: **for** $j \leftarrow 1 : size\,(z_t)$ **do** ▷ Correction step

6: find m such that z_t^j refers to map landmark m ▷ Correspondence

7: **if** landmark m never seen before **then** ▷ Initialize new feature

8: $\mu_{m,t}^{[p]} \leftarrow h^{-1}\left(z_t^j, x_t^{[p]}\right) \quad \Sigma_{m,t}^{[p]} \leftarrow H^{-1} Q_t \left(H^{-1}\right)^T$

9: $w_t^{[p]} \leftarrow w_t^{[p]} p_0$ ▷ Weight update with a default value

10: **else** ▷ Update existing feature

11: $\bar{z} \leftarrow h\left(x_t^{[p]}, \mu_{m,t-1}^{[p]}\right), \quad Q \leftarrow H \Sigma_{m,t-1}^{[p]} H^T + Q_t$

12: $K \leftarrow \Sigma_{m,t-1}^{[p]} H^T Q^{-1}$ ▷ Kalman gain

13: $\mu_{m,t}^{[p]} \leftarrow \mu_{m,t-1}^{[p]} + K\left(z_t^j - \bar{z}\right), \quad \Sigma_{m,t}^{[p]} \leftarrow (I - KH)\,\Sigma_{m,t-1}^{[p]}$

14: $w_t^{[p]} \leftarrow w_t^{[p]} \frac{1}{\sqrt{2\pi Q}} e^{-\frac{1}{2}\left(z_t^j - \bar{z}\right)^T Q^{-1}\left(z_t^j - \bar{z}\right)}$ ▷ Weight update

15: **end if**

16: **end for**

17: **for** all features m not existing in z_t **do**

18: $\mu_{m,t}^{[p]} \leftarrow \mu_{m,t-1}^{[p]}, \quad \Sigma_{m,t}^{[p]} \leftarrow \Sigma_{m,t-1}^{[p]}$

19: **end for**

20: **end for**

21: **for** $p \leftarrow 1 : size\,(Y_{t-1})$ **do** ▷ Normalization of the weights

22: $\bar{w}_t^{[p]} \leftarrow \frac{w_t^{[p]}}{\sum_{m=1}^{M}\left(w_t^{[m]}\right)}$

23: **end for**

24: $N_{eff} \leftarrow \frac{1}{\sum_{m=1}^{M}\left(\bar{w}_t^{[m]}\right)^2}$

25: **if** $N_{eff} < N_{thresh}$ **then** ▷ Selective resampling - see Appendix B

26: $Y_t \leftarrow resampler\left(w_t^{[1:size(Y_{t-1})]}, x_t^{[1:size(Y_{t-1})]}, \left(\mu_{1,t}^{[1:size(Y_{t-1})]}, \Sigma_{1,t}^{[1:size(Y_{t-1})]}\right)\right)$

27: $W_t \leftarrow [1, 1, ..., 1]_{1xSIZE(Y_t)}$

28: **else**

29: $Y_t \leftarrow \langle x_t^{[1:size(Y_{t-1})]}, \left(\mu_{1,t}^{[1:size(Y_{t-1})]}, \Sigma_{1,t}^{[1:size(Y_{t-1})]}\right)\rangle$

30: $W_t \leftarrow w_t^{[1:size(Y_t)]}$

31: **end if**

Chapter 3

Navigation using stereo vision or LiDAR technology

3.1 The LiDAR technology

LiDAR stands for Light Detection And Ranging system[12] and it is a technique for measuring distances using laser pulses. More in detail, the LiDAR sensor emits laser pulses which are reflected on the nearby objects and return to it. By measuring the round-trip time, it is easy to compute the distance from the objects by using the formula:

$$distance = \frac{c\, t_{rt}}{2}$$

with c being the speed of light and t_{rt} the round-trip time of the laser pulse.

Depending on the situation, a LiDAR sensor may measure one or more distances, depending on the number of the returned pulses for each of the emitted ones (e.g. through a tree, each leaf will reflect some photons, until the pulse reaches to a branch or the ground which will reflect the remaining). The LiDAR sensor constructs a waveform with the received pulses. Then, it detects its peaks (which correspond to the returned pulses) by using a threshold value and, finally, it computes the distances by comparing the time when those peaks were received with the time when the pulse was transmitted (see figure 3.1).

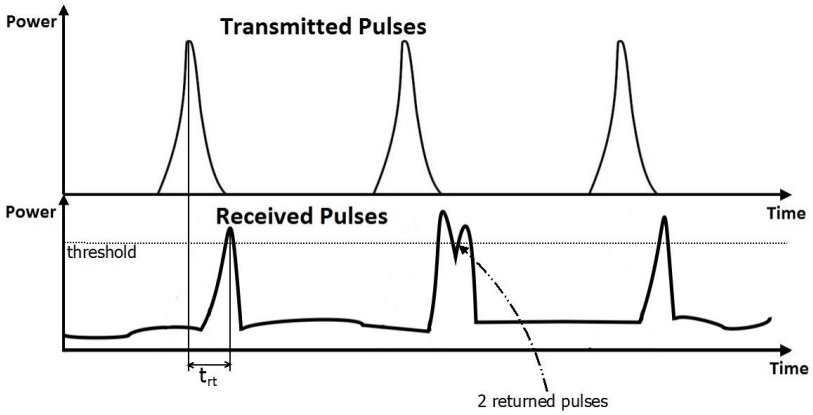


Figure 3.1: LiDAR Transmitted vs Received Waveforms.

A great variety of LiDAR sensors exist with a range from a few to hundreds of meters. Most of them are 2D (i.e. they only scan a flat wide-angle 'slice' of

the environment, e.g. figure 3.2) and provide their measurements in pairs of the form $(distance, angle)$ with the angle indicating the direction of the measurement relatively to the reference axis of the sensor. 3D sensors (i.e. they scan all the dimensions of the environment) also exist and they provide their measurements in a point cloud form[13].

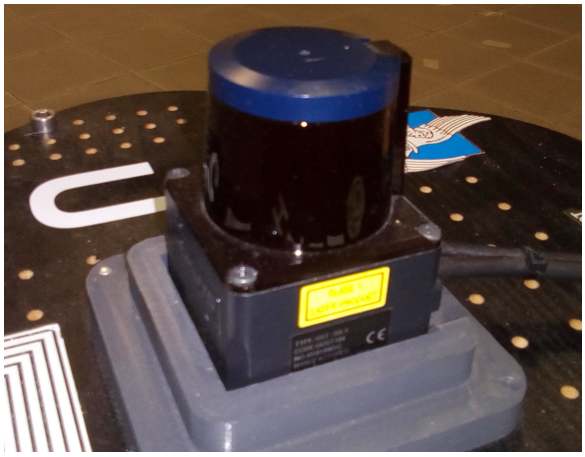


Figure 3.2: Wide-angle 2D LiDAR sensor.

3.2 The stereo vision technology

Stereo vision technology is used to produce 3D images (with x,y,z being the width, height, depth respectively). In contrast to the common cameras, a stereo camera needs to capture two individual images of the same sight but from different points of view (for example, a stereo camera may consist of two common cameras at a fixed distance and a mini computer that will process their images and will produce the 3D image). In what concerns the processing of those two images for the 3D image to be produced, the stages are the following[14]:

1. **Image rectification:** The transformation of both the left and the right image so that they are coplanar (i.e. projected on the same plane). It would be useless in the ideal case where the left camera was only horizontally displaced by an offset, compared to the right one (i.e. not moved towards an object or rotated). However in general the 2 images are not projected on the same plane (perfect coplanarity is hard even with high-accuracy equipment), thus rectification is usually performed. Epipolar curve is defined as the curve which corresponds to the area of view of the first camera inside the image captured by the second camera. After the rectification, the matching pixels from the two images have the same y coordinate (same row in the images), which makes their correspondence easier. Thus, the epipolar curves turn into horizontal epipolar lines.

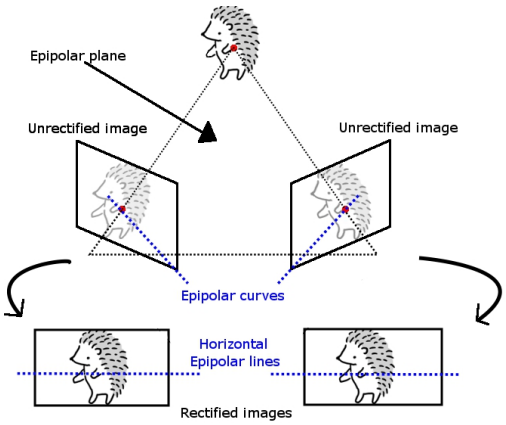


Figure 3.3: Rectification of 2 stereo images.

2. **Image matching:** The purpose of this step is to correspond each pixel of the one image to one on the other (since the 2 images capture the same sight but from different views). The matching algorithm computes the similarity between each pair of pixels and determines the most probable correspondence. When the images are rectified, that process is even simpler because the search space for each pixel is restricted to the corresponding horizontal line on the other image (its epipolar curve).

3. **Exclusion of unique areas:** Due to the different viewpoints of the 2 cameras, each image may contain an area not visible by the other one. Those areas have to be excluded from the 3D image that is to be produced, because the computation of the disparity/depth for those areas is not possible. A left-right consistency check can be used between 2 separate disparity maps (one for each image) and the pixels whose disparities do not coincide to be excluded.

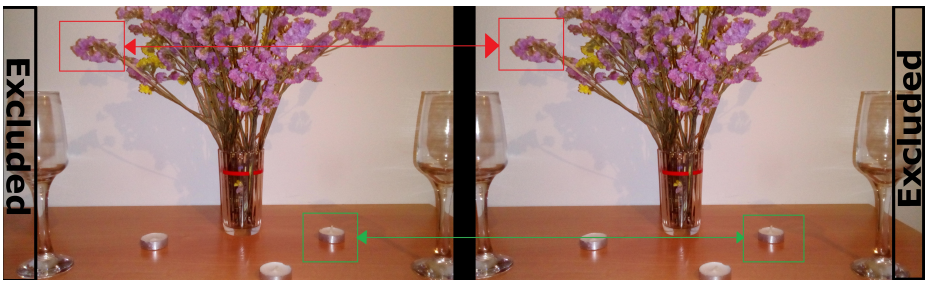


Figure 3.4: Matching and exclusion of unique areas for 2 rectified stereo images.

4. **Construction of a Disparity Map:** Disparity is the (horizontal, for rectified images) displacement of each pixel of the first image, from its corresponding to the second one. After the computation of all the disparity values, the disparity map is induced.

5. **Computation of the depth image:** Finally, the z (depth) dimension of the produced 3D image can be computed. More specifically, depth is inversely proportional to the disparity value of each pixel: $depth = \frac{Bf}{disparity}$ with B the distance between the two cameras and f the focal length of the cameras.

All in all, by following the above procedure a 3D image is produced. That image, except from the view that it represents, it also contains information about the distance of the cameras from the objects captured in the image, which can be used as input to navigation algorithms (e.g. localization, SLAM)

3.3 The ROS Framework

The Robot Operating System (ROS) is a middleware developed by the 'Willow Garage' Research Lab and released in 2010, with a purpose to simplify the interaction between different devices. By hiding the details of each device's specific API, it achieves a higher-level communication between them without the need of deepening into too much low-level details. Thus, each implementation is independent of the hardware and can be used in any ROS-compatible system. ROS consists of various components (whose relations can be seen in figure 3.5), the most important of which are:

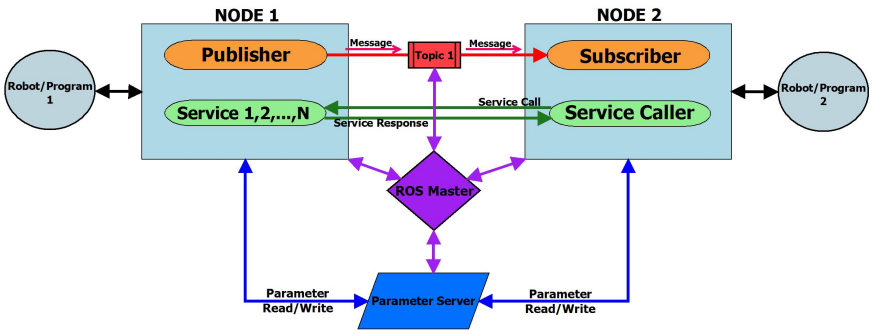


Figure 3.5: Communication inside the ROS framework.

- **ROS Master:** Master is the core of ROS, responsible for naming and registering new nodes, as well as matching publishers/subscribers to topics/services. Moreover, it controls the Parameter Server. Thanks to the Master the nodes track each other and can start peer-to-peer communication anytime it is needed.
- **Nodes:** Each node is a piece of code executing tasks (e.g. communicating with an actual/simulated robot, sending commands).
- **Topics:** Topics are the "channels" through which the nodes communicate. ROS uses the publisher-subscriber technique, with each node publishing its output data and subscribing at topics from which it reads its input data. This is a one-way communication from the perspective that there is no way of sending a response to the publisher of the data.

- **Messages:** Information exchanged between nodes comes in the form of messages, which are transmitted through the topics.
- **Services:** In contrast to topics, services offer a two-way communication in ROS. When a node offers a service, it is publicly available to the rest of the nodes and anyone can call it and wait for the returned result (just as a function call inside a code).
- **Parameters:** This is a way to share static configuration data between all nodes. All parameters are kept in the Parameter Server, from where any node can read or save information.

3.4 Localization with ROS algorithms and optical sensors

There are many, implemented in the ROS framework, algorithms for the purposes of localization. Most of them require distance metrics as measurements from the landmarks in the environment, thus a LiDAR or the data from a depth image is ideal. 'AMCL', 'AMCL3D', 'robot_localization', 'mrpt_localization' are examples of such packages, each one with both advantages and disadvantages. However, a thorough analysis of their specific details is not a purpose of this thesis, thus experiments were not conducted additionally to the brief theoretical study.

3.5 SLAM with the 'Google Cartographer'

There is a great variety of SLAM algorithms implemented as packages inside the ROS framework. Some of them are implementations of the well-known strategies mentioned in section 2.2, while others use extra tricks and variations to increase their efficiency.

A detailed analysis of all the existing algorithms is out of the purposes of this thesis, which attempts a summary of the fundamentals in indoor navigation and studies more in-depth an approach of improving the RFID-based localization.

One of the most popular and efficient algorithms is the 'Google Cartographer' [15]. It mainly consists of a local and a global SLAM subsystem executed simultaneously in separate threads. Local SLAM constructs sub-maps with collecting data from the sensors, discarding some and keeping the rest, and assembling area scans from which it constructs each map. Global SLAM is needed to combine the sub-maps produced by the local SLAM in order to construct the entire map. It also solves the 'Loop-Closure' problem, i.e. the problem of recognition of the same map features when they are seen for a second time (and they are not clearly identifiable). To achieve that, a graph is constructed with its nodes being possible sub-map correspondences and its edges containing constraints. Thus, an optimization problem is induced on the graph whose solution is the desired map and robot's pose. Two maps produced by this algorithm are given in section 5.1.

Chapter 4

Navigation using RFID tags

4.1 The RFID technology

Radio-frequency identification (RFID)[16] is a technology used to identify a tag using radio waves emitted by the reader's antenna. The tag may be located far from the reader, not essentialy in a line of sight.

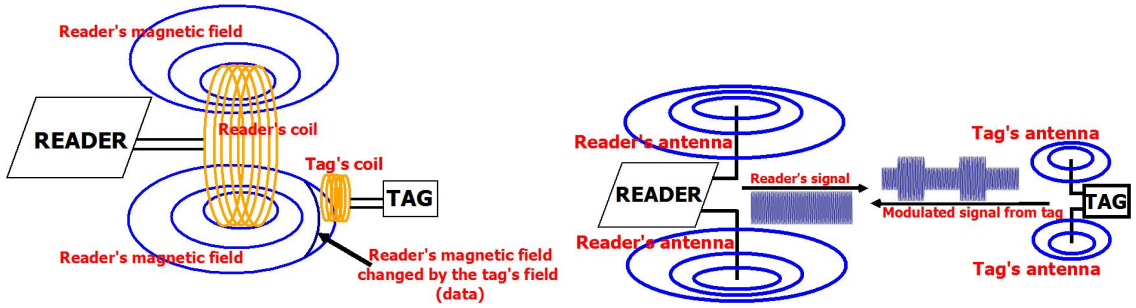
4.1.1 The RFID architecture

Tags can be separated in active and passive ones. The active tags require an external power source in order to operate, while the passive ones power up from the energy stored in the reader's emitted signal (varying from $10\mu W$ to $1mW$). The reasons for which the latter ones gain more popularity are obvious; they are related with both the price and the usefulness of the passive tags, as they can be attached to and used for the identification of any kind of objects. On the other hand, the power level of the signal emitted from the reader has to be significantly higher comparing to the one required for the active tags.

Passive tags consist of 3 main components: an antenna, an integrated circuit connected to it and a substrate on which the former two are attached. Finally, the design may be encapsulated inside a label or any other components. There are two different ways to transfer energy from the reader to the tag, which exploit the electromagnetic properties of an RF antenna:

- **Near-Field Coupling (NFC)**: This method uses the principles of magnetic induction with both the reader and the tag having a coil. The reader creates an alternating magnetic field in an area close to its antenna. If the tag is placed inside that field, an alternating voltage appears to the tag's coil. After the rectification and coupling of that voltage to a capacitor, an amount of charge is available to power up the chip of the tag. The tag's data will be transmitted to the reader through the technique of *load modulation*: the reader will detect the slight changes to its magnetic field which occur from the currents on the tag's coil (and they will oppose the reader's field) (see fig. 4.1a). However, this method can be used only for short distances, and more specifically distances smaller than the wavelength of the reader's radiation, due to the physical limitations of that method. More disadvantages of that method are the high transmission energy and the low data transfer rate (for the tag-reader link).

- **Far-Field Coupling (Backscattering)**: This method harvests energy from the electromagnetic waves transmitted by the reader's antenna. The reader and



(a) Near-Field Coupling. (b) Far-Field Coupling.

Figure 4.1: Ways of reader-tag communication.

the tag have a dipole antenna each. When the tag receives the reader's waves an alternating voltage appears to its antenna. It is rectified and linked to a capacitor, so that it ends up being an amount of charge that will power up the tag. The remaining energy of the radio waves is modulated with the tag's data and be sent back to the reader (see fig. 4.1b). This method can be used in longer distances than those possible in NFC.

For the purposes of a vehicle's navigation, the most efficient solution is that of the **passive backscattering** tags (and this is the kind of tags that will be examined further below). Passive tags can be easily located at any point without taking care for a power supply to each one of them. Moreover, the NFC method is not practical due to the need for movement of the vehicle around the free space without the strict limitation of a very short distance from the tag.

4.1.2 The RFID measurements: EPC, RSSI, Phase

After each communication between the reader and the tag, there are three pieces of information available to the reader: the EPC, the RSSI and the Phase.

- **Electronic Product Code (EPC)**: This alphanumeric code is the identity of each tag. More specifically, EPC is the data transmitted from the tag to the reader (by modulating the reader's signal, as explained in Section 4.1.1.

- **Received Signal Strength Indication (RSSI)**: This is the power of the received (modulated) signal coming from the tag. The way in which the RSSI will be modeled is described in a previous work [7], with two modifications. First, the multistatic model has to be converted in a monostatic one (as the emitter is the same with the reader in our case), and second the 2D localization has to be adapted for the 3D space.

  Finally, it occurs (see proof in appendix A) that the received power equals to: $y_{out} = P_T n_T G_R^2 G_T^2 L^2 |h|^4$, where:

  - P_T is the transmission power of the reader.
  - n_T is the tag scattering efficiency.

- G_R is the gain of the reader's antenna.
- G_T is the gain of the tag's antenna.
- L models the path loss between the reader and the tag. For simple environments (i.e. few surfaces for the signal to be reflected), the 2-ray model (modeling the direct-path ray and the reflection from the ground) is accurate enough (but for more complex environments it is just an approximation). If $D = dist(reader, tag)$ and h_R, h_T the heights (from the ground) of the reader and the tag respectively, then:

$$L = \begin{cases} \left(\frac{\lambda}{4\pi D}\right)^2, & D < \frac{4\pi h_R h_T}{\lambda} \\ \left(\frac{h_R h_T}{(D)^2}\right)^2, & D \geq \frac{4\pi h_R h_T}{\lambda} \end{cases}$$

- $|h|^4$ is the square of a Gamma random variable, meaning $|h|^2 \sim \Gamma(k, \theta)$. This variable models the effect of the multipath (i.e. reflections of the radiation on surfaces of the area e.g. walls are combined and create a different than the expected signal to be received by the reader) in the environment, which is Gamma distributed for each one of the 2 paths (reader→tag, tag→reader).

- **Phase of the received signal**: The phase of the (modulated) signal received by the reader depends on its distance from the tag and equals to $\phi_{out} = -\frac{4\pi D}{\lambda} + \theta + \phi_n$ (with D being the tag-reader distance, θ being a constant offset owed to the length of the reader's and the tag's wires and ϕ_n a changing offset due to the multipath).

  However, the reader returns a value in $[-\pi, \pi)$ or $[0, 2\pi)$, which means that $\phi_{reader} = \phi_{out} \bmod 2\pi$. That gives many candidate distances instead of just one and makes essential the use of an inference method to determine the most probable.

All in all, both the RSSI and the phase measurements can be used in a localization algorithm. However, in terms of accuracy multipath affects the RSSI significantly more than it does with the phase measurements. As a result, in environments with strong multipath the variance of the RSSI is much greater than that of the phase.

4.2 Existing work

Enough methods have been proposed for localization and SLAM using RFID tags. Some of them use the RSSI (when the accuracy is not too important), while others use the phase in order to make a more accurate estimation (just few centimeters error). On the other hand, the algorithms using the phase have a higher complexity than those using the RSSI, because of the great number of distances which correspond to each phase value (and the need to discover the correct one, additionally to reducing the noise's impact). For the needs of a vehicle's navigation the use of the phase is necessary, since there is a need for high-accuracy estimation, with some of the better approaches given below.

4.2.1 Localization using RSSI, Phase and Kalman Filters

The first approach[5] to be examined solves both an active (the robot's motion is used) and passive (immovable objects) localization problem using Kalman Filters. The robot has an RFID reader and moves inside the environment where anchor tags are located in fixed positions. However, the locations of the tags are not precisely known, but some uncertainty is involved. While time flows, the location of both the robot and the tags is estimated with increasing accuracy. Thus, this is not exactly a localization problem, but more likely a SLAM problem with some initial knowledge available (i.e. the tags are not discovered while the algorithm runs, but they are known in advance).

Two variations of Kalman Filter are used by this algorithm. An Extended Kalman Filter (EKF) utilizes the Taylor series to approximate non-linear functions around a Gaussian curve and use that Gaussian in order to fuse observations and predict the future. In other words, it linearizes a non-linear model and then, it behaves as an ordinary Kalman Filter. However, depending on the model this linearization may be a bad approximation and the EKF might become unstable or converge very slowly. Unscented Kalman Filters (UKF) solve that issue by approximating the model without linearizing it first (better approximation of the model, thus more quick convergence).

The proposed algorithm utilizes both of these filters the way explained below. Actually, it handles many filter instances concurrently with a different initialization point (state vector) each. Weights are assigned to them, and while time passes the instances with low weights are deleted, until only one remains. Anytime in the case of a low likelihood value for the remaining instances, the algorithm restarts. Initially all instances "work" on the first stage, and when they satisfy the transition criterion they proceed to the second stage. The only case to go back to the first stage is the algorithm to be restarted. The transition from the first to the second stage happens for each part of the state vector (tag/robot coordinates) and for each instance independently. The criterion is the variance of the respective estimate; when it falls under a threshold (i.e. the estimation is close to the real state) stage 2 is used to further increase the accuracy.

1. On the **first** stage, the robot's motion is fused with the RSSI observations inside an Unscented Kalman Filter (UKF). The state vector of the filter contains the

coordinates of the robot, as well as the coordinates for each one of the tags and it is accompanied by a covariance matrix. At each time step the motion (i.e. the data from the robot's IMU) is fused along with the state vector and the a priori estimate (state vector + covariance matrix) is computed. Afterwards, this estimate is corrected by the RSSI measurements taken at the same time step and the final estimate is returned. The choice of a UKF is based on the very complex non-linear model of the RSSI measurements (for which an EKF would be unsuitable).

2. On the **second** stage, both the RSSI and the phase are used. The RSSI and the motion of the robot are fused as previously inside an Unscented Kalman Filter. Then, the estimation returned by the UKF is fused with the phase observations inside an Extended Kalman Filter (EKF). EKF is preferred in this case as the algorithm's estimate is close to the real one now, thus the linearization performed by the EKF is a good enough approximation.

4.2.2 Localization using Phase and a Particle Filter

The second approach[6] is a passive localization algorithm with as few parameters as possible, designed to investigate the accuracy that can be achieved in various environments. Its purpose is the localization of an immovable tag by using only the phase measurements (by a moving robot whose trajectory is precisely known), and they are integrated inside a particle filter. The weights of the particles are computed based on a distance metric (instead of the actual phase) so that slight changes of phases near to $0/2\pi$ due to noise do not influence the weights significantly. The method computing the weight update of the particles is explained more thoroughly in section 4.3.2.

4.2.3 SLAM using Phase and a Particle Filter

The last of the approaches[4] to be analyzed is about the SLAM problem using RFID tags. Again, only the measured phase is used and the FastSLAM algorithm as described in section 2.2.2.

The special characteristic of this algorithm is owed to the nature of the phase values, which correspond to multiple distances each. When a new tag is detected, a Gaussian instance for it has to be initialized in every particle. However the possible coordinates lie on multiple spheres (for 3D SLAM, or circles for 2D) and just one instance is not enough. Thus, a set of instances is initialized for each newly detected tag in multiple angles and distances. While more and more observations are integrated inside the filter, the instances with very low weights are deleted. Finally, the state estimation at each time step is the weighted average of all the remaining instances and all the particles.

Apart from the difference in the nature of the problems (localization vs SLAM), there are two major differences in this algorithm compared to 4.2.2. First, this approach is more complex, since it implements two particle filters and one Kalman filter

compared to 4.2.2 where only one particle filter is used. Second, the weight update of the particles takes into account the difference in the phases and not the distance that they represent, as in 4.2.2. Consequently, phases close to 0 and 2π may not have similar weights, despite corresponding to almost equivalent distances.

4.3 Localization using RSSI and Phase inside a particle filter

The idea of this approach is that for the fusion of RSSI and phase measurements, a particle filter is a good approach (and easy-to-implement at the same time), if the right models are used for the weight updates. The purpose is for the RSSI to be used in a way which will further improve the accuracy succeeded by the phase measurements.

A passive (with respect to the robot's motion), single-robot, global localization scenario in a static environment is used to test the efficiency of the approach. The specific scenario was chosen due to its simplicity, in contrast to active localization approaches or SLAM problems. Moreover, change of accuracy in this scenario would result to a respective accuracy change in the rest scenarios too.

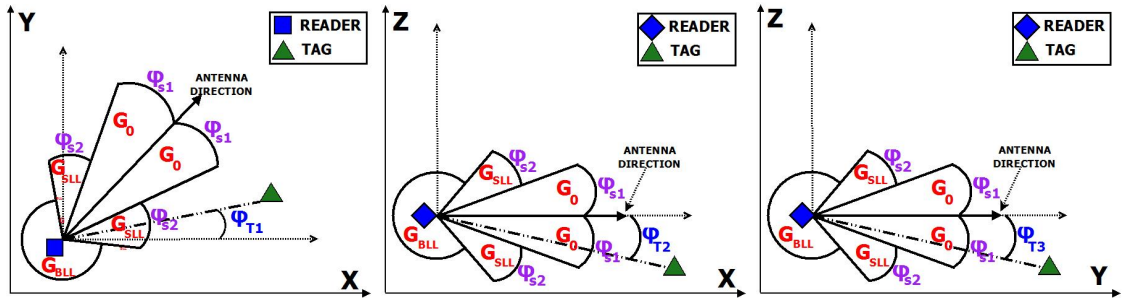
More in detail, the problem is the one of the localization of an immobile tag (passive localization). The RFID reader is located on a moving robot, whose trajectory is known (thus, the exact location of the reader is also known for each observation). Each observation consists of an RSSI and a phase value which will update the weights of a particle filter. Localization of more than one tags simply requires the execution inside a loop. In appendix F an example of a more complex localization algorithm is given (localization of a moving robot using anchor tags), where the robot's motion model is also used. However, no further analysis or experimental results are conducted, due to its greater number of parameters which need tuning compared to the previously mentioned approach.

Algorithm 6 is used, which is a common Monte-Carlo Localization algorithm. The resampling method is optimized as described in appendix B. Lines 12 & 13 are of particular interest, as they are induced from the models of the RSSI and the phase. More details and explanation follow.

4.3.1 RSSI Weight Update

This section explains the way that the W_{RSSI} weight update is computed for a particle $X^{[m]} = \left[x^{[m]} y^{[m]} z^{[m]}\right]$ that represents the tag's location, a vector $g = [x_g \ y_g \ z_g \ \theta_g]$ symbolizing the reader's location & orientation and a RSSI measurement y_{out}.

The model described in section 4.1.2 is used. What is more, the gain of the reader's antenna is not a constant, but depends on the signal's angle of arrival because the antenna is directional. A simplified model is used, described below:



(a) X-Y plane. (b) X-Z plane. (c) Y-Z plane.

Figure 4.2: 3D model of directional antenna.

3D Antenna Model:

A 3D directional antenna model is used, as demponstrated in figure 4.2. X-Y plane is the ground and the Z axis is vertical to it. The 3-dimensional antenna vector is always vertical to the z axis (parallel to the ground). The gain of the main lobe equals with G_0, and the gain of the side lobes with G_{SLL}. Finally, the backside lobe has a gain of G_{BLL}. Thus, the reader's antenna gain at the direction $(\phi_{T1}, \phi_{T2}, \phi_{T3})$ of the tag is given by:

$$G_R(\phi, \phi_{T1}, \phi_{T2}, \phi_{T3}) = \begin{cases} G_0, & all(\phi_{T1}, \phi_{T2}, \phi_{T3}) \in [\phi - \phi_{s1}, \phi + \phi_{1s}] \\ G_{BLL}, & \phi_{T1} \; or \; \phi_{T2} \; or \; \phi_{T3} \notin [\phi - \phi_{s1} - \phi_{s2}, \phi + \phi_{s1} + \phi_{s2}] \\ G_{SLL}, & elsewhere \end{cases}$$

By making a simplification, $\phi_{s1} = \phi_{s2}$, i.e. the main lobe is assumed to have opening of angle $2\phi_s$, twice of the two side lobes which have opening of angle ϕ_s each.

Computation of the weight update:

According to section 4.1.2 $RSSI = P_T n_T G_R^2 G_T^2 L^2 |h|^4$ with:

- $G_R = G_R(X^{[m]}, g) = G_R^{[m]}$
- $L = L(X^{[m]}, g) = L^{[m]}$.
- $|h|^2 \sim \Gamma(k, \theta)$

If it is denoted that $a^{[m]} = P_T n_T \left(G_R^{[m]}\right)^2 G_T^2 \left(L^{[m]}\right)^2$, then $RSSI = a^{[m]}|h|^4$. It can be shown (Appendix C) that, for a RSSI measurement y, the weights of the particles are updated as follows:

$$W_{RSSI} = f_{y|X^{[m]}}\left(y|X^{[m]}\right) = \frac{1}{2\sqrt{a^{[m]}y}} \frac{\left(\left(\frac{y}{a^{[m]}}\right)^{\frac{k-1}{2}}\right)\left(e^{-\frac{1}{\theta}\sqrt{\frac{y}{a^{[m]}}}}\right)}{\theta^k \Gamma(k)} =$$

$$= \frac{1}{2\theta^k \Gamma(k) \left(a^{[m]}\right)^{k/2}} (y)^{\frac{k-2}{2}} \left(e^{-\frac{1}{\theta\sqrt{a^{[m]}}}\sqrt{y}}\right)$$

Estimation of the (k, θ) parameters for the Gamma distribution:

The estimation method used is the Maximum Likelihood (ML) estimation. As proved in appendix D, given a set of N independent observations $\{y_1, y_2, ..., y_N\}$ with $y_i = a_i z^2$, $z \sim \Gamma(k, \theta)$, and the respective values of the scaling variable a $\{a_1, a_2, ..., a_N\}$ the estimated parameters are computed as follows:
$\hat{\theta} = \frac{1}{Nk} \sum_{i=1}^{N} \sqrt{\frac{y_i}{a_i}}$, $\hat{k} \approx \frac{3+\sqrt{9+12s}}{12s}$ with $s = \ln\left(\frac{1}{N} \sum_{i=1}^{N} \sqrt{\frac{y_i}{a_i}}\right) - \frac{1}{N} \sum_{i=1}^{N} \ln\left(\sqrt{\frac{y_i}{a_i}}\right)$
However due to the nature of the particle filters, two problems arise:

1. The value of the $\{a_i\}$ scaling variables is unknown (since they depend on the tag's location). Moreover, these parameters cannot be included inside the particles, as they are used for the computation of their weights. Their inclusion would require the use of the Expectation - Maximization Algorithm, which does not guarantee the finding of a global maximum (i.e. the right location of the tag).

   To face that problem an anchor tag can be used. Since its location is known, the $\{a_i\}$ parameters are also known and k, θ can be easily computed. It is important to note that they are considered constant inside the environment so that the resulting model is simplified (despite the fact that this assumption is not accurate, especially for complex environments).

2. At the initial time steps, the observations are too few for a correct estimation of k, θ. Thus, a fixed predetermined value has to be used, until their number exceeds a certain limit (see fig. 1). Afterwards, they are computed according to the formulas mentioned above. Of course in some scenarios this issue does not even exist, since the robot may have already estimated the parameters before it starts localizing a tag.

4.3.2 Phase Weight Update

Let $X^{[m]} = \left[x^{[m]} y^{[m]} z^{[m]} \theta^{[m]}\right]$ be a particle corresponding to the tag's location and constant phase offset, $g = [x_g \ y_g \ z_g \ \theta_g]$ the reader's position and orientation and ϕ_{out} the phase measurement returned by the reader (getting values in the interval $[0, 2\pi)$ or $[-\pi, \pi)$).

The particle's weight update $W_{PHASE} = p\left(\phi_{out} | X^{[m]}\right)$ for that measurement is based on the model described in 4.1.2. Given a phase, the candidate points for the tag's position are located on concentric spheres centered on the reader and with radius difference equal to $\lambda/2$. The value of the phase determines the radius of the smallest sphere ρ_ϕ and it is compared with the respective spheres as they should be based on the particle-reader distance (fig. 4.3).

This method[6] converts phase into a distance metric before it computes $p\left(\phi_{out} | X^{[m]}\right)$. That is very important in environments with multipath, because the noise may change a value near to 0 into a value near to 2π and vice versa. Thus, the weights for both values should be relatively close to each other.

▷ Subtraction of a constant offset from all phase observations

1: $\phi^* = ((\phi_{out} \bmod (2\pi)) - \theta^{[m]}) \bmod (2\pi)$

▷ Computation of the radius for the 2 spheres to be compared

2: $\rho_\phi = \frac{\lambda\phi^*}{4\pi}, \qquad \rho_d = ||X_{1:3}^{[m]} - g_{1:3}||_2 \bmod (\lambda/2)$

3: $\rho_{max} = max(\rho_\phi, \rho_d), \qquad \rho_{min} = min(\rho_\phi, \rho_d)$

▷ Distance of the observation from the nearest peak in the phase's distribution

4: $\Delta^{[m]} = min(\rho_{max} - \rho_{min}, \ \frac{\lambda}{2} - (\rho_{max} - \rho_{min}))$

5: $W_{PHASE} = p(\Delta^{[m]}|X^{[m]}) \equiv N(\Delta^{[m]}; 0, \sigma^2_{phase})$ ▷ Weight computation

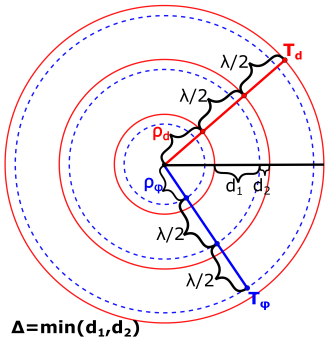


Figure 4.3: Conversion of a phase observation to distance metric.

Note: σ_{phase} is the standard deviation of the phase measurements and is computed heuristically ($\simeq 0.05\lambda - 0.2\lambda$, depending on the amount of multipath).

4.3.3 Resampling

When the object that is to be localized does not move, resampling should not happen as described in appendix B. That's because the duplicate particles are useless, since they will have exactly the same weights without ever changing location. To face that problem, a Gaussian approximation of the resampled distribution is used and the particles are drawn from that. This way, the probability of two particles to be sampled and be equal is almost zero.

Another issue is the value of the threshold that determines the frequency of the resampling, with a commonly used value being $N_{thresh} = \frac{M}{2}$. However, there is a high risk of particle deprivation and the threshold has to be set to a significantly lower value. The reason is that the new set of particles is drawn from a Gaussian approximation which is unimodal, while the distribution of the particles is initially multimodal (since a phase observation corresponds to multiple distances). As time passes, the multimodal distribution gradually turns into a unimodal one by converging to the right mode but if resampling happens too often, particles may be deprived from it.

Algorithm 6 Tag localization

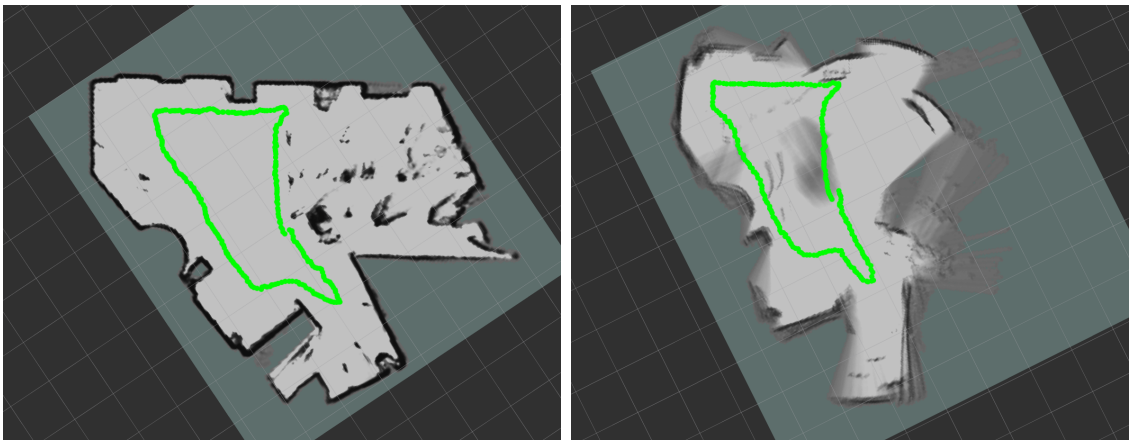
Set M (number of particles), T (time window), N_{thresh}
Set $X^a_{min}, X^a_{max}, Y^a_{min}, Y^a_{max}, Z^a_{min}, Z^a_{max}$(search area dimensions)
Set $\{g^{[1]}, g^{[2]}, ..., g^{[T]}\}$ (the locations of the robot for time t=1,2,...,T)
for m$\leftarrow$ 1:M **do**
    $x^{[m]} \sim U\left[X^a_{min}, X^a_{max}\right], y^{[m]} \sim U\left[Y^a_{min}, Y^a_{max}\right], z^{[m]} \sim U\left[Z^a_{min}, Z^a_{max}\right], \theta^{[m]} \sim U\left[0, 2\pi\right)$
    $X_0^{[m]} \leftarrow \left[x^{[m]} y^{[m]} z^{[m]} \theta^{[m]}\right], w^{[m]} \leftarrow 1$
end for
for t$\leftarrow$ 1:T **do**
    Collect the RSSI y_t and the phase ϕ_t.
    for m$\leftarrow$ 1:M **do**
        for j$\leftarrow$ 1:G **do**
            $W_{RSSI} \leftarrow f_{y_t|X_t^{[m]}}\left(y_t|X_t^{[m]}\right) = \frac{1}{2\theta^k \Gamma(k)\left(a_t^{[m]}\right)^{k/2}} \left(y_t\right)^{\frac{k-2}{2}} \left(e^{-\frac{1}{\theta\sqrt{a_t^{[m]}}}\sqrt{y_t}}\right)$
            $W_{PHASE} \leftarrow p(\Delta_t^{[m]}|X_t^{[m]}) \equiv N(\Delta_t^{[m]}; 0, \sigma^2_{phase})$
            $w^{[m]} \leftarrow w^{[m]} W_{RSSI} W_{PHASE}$ ▷ Correction step
        end for
    end for
    $w^{[1:M]} \leftarrow \frac{w^{[1:M]}}{\sum_{m=1}^{M} w^{[m]}}$ ▷ Normalization of weights
    $N_{eff} \leftarrow \frac{1}{\sum_{m=1}^{M}\left(w^{[m]}\right)^2}$
    $\hat{X}_{est}^{[t]} = \sum_{m=1}^{M} w^{[m]} X_t^{[m]}$ ▷ Position estimation for time t
    if $N_{eff} < N_{thresh}$ **then** ▷ Resampling if needed
        $X_{LVS}^{[1:M]} \leftarrow LVS\left(X_t^{[1:M]}, w^{[1:M]}\right)$
        $X_{t+1}^{[1:M]} \sim N\left(E\left[X_{LVS}^{[1:M]}\right], Var\left[X_{LVS}^{[1:M]}\right]\right)$
        $w^{[1:M]} \leftarrow 1$
    else
        $X_{t+1}^{[1:M]} \leftarrow X_t^{[1:M]}$
    end if
end for
return $\hat{X}_{est}$

Chapter 5

Numerical results

5.1 SLAM maps by the 'Google Cartographer'

In figure 5.1 two maps of the same place are given, the one constructed with a LiDAR sensor and the other with a stereo camera (green color denotes the algorithm's estimation for the path of the robot). Due to the camera's narrow angle-of-view and its lower accuracy, the map produced by it is not as accurate as the map by the LiDAR.



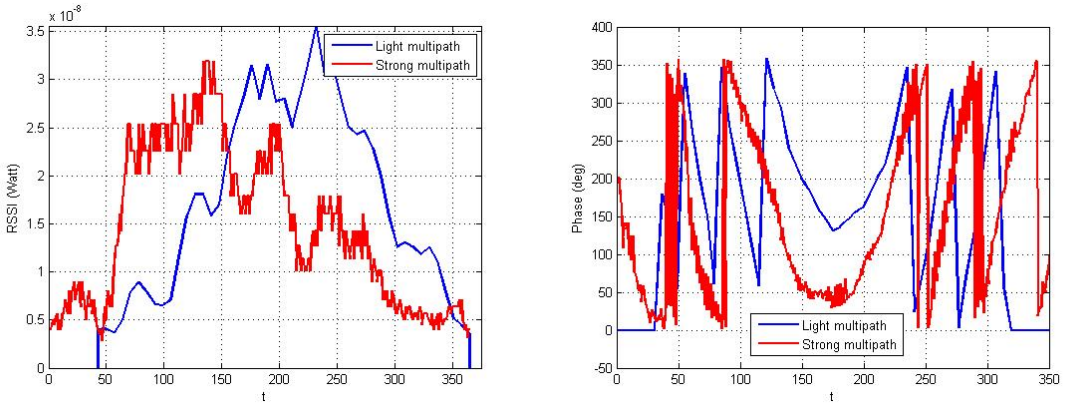
(a) Map by using a LiDAR sensor. (b) Map by using a stereo camera.

Figure 5.1: Maps constructed by Google Cartographer.

5.2 Accuracy of RFID-based localization algorithms

In Figure 5.2 the RSSI and the phase measurements are plotted, exactly as returned by the RFID reader. Relatively with the robot's trajectory, it is a straight line with the distance from the tag initially being decreased and afterwards being increased. It is worth noting that the noise (because of the multipath) affects the RSSI significantly more than the phase. Moreover, the plot of the RSSI should be unimodal (according to the squared Gamma model used in this work), but the greater the amount of multipath that exists, the more modes to also appear.

The choice of the location for the anchor tag is important. To simplify the model, it has been assumed that the parameters (k, θ) of the RSSI's Gamma distribution do



(a) RSSI observations. (b) Phase observations.

Figure 5.2: RSSI & Phase measurements inside different environments.

not change from point to point, but the reality is quite different. Thus, the actual parameters are not exactly the same with the ones estimated by the anchor tag.

Figure 5.3 demonstrates how the anchor tag's location affects the RMSE of the estimation. Each curve corresponds to a different distance between the anchor tag and the tag which has to be localized, and the particle filter uses only the RSSIs in order to update the weights of the particles. The wavelength of the reader equals with 0.173 m and the search area's dimensions with 2m x 3m x 2m. As expected, the closer that the anchor is to the unknown tag, the better the estimation of the parameters is, and the lower the RMSE is too. When that distance is greater than $\lambda/2$, then the tags are uncorrelated and the estimation of the parameters is not accurate. On the other hand, distance less than half of the wavelength induces correlated tags and a good estimation.

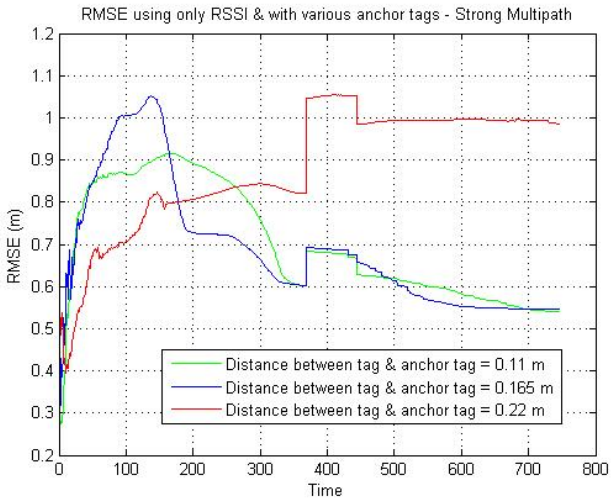
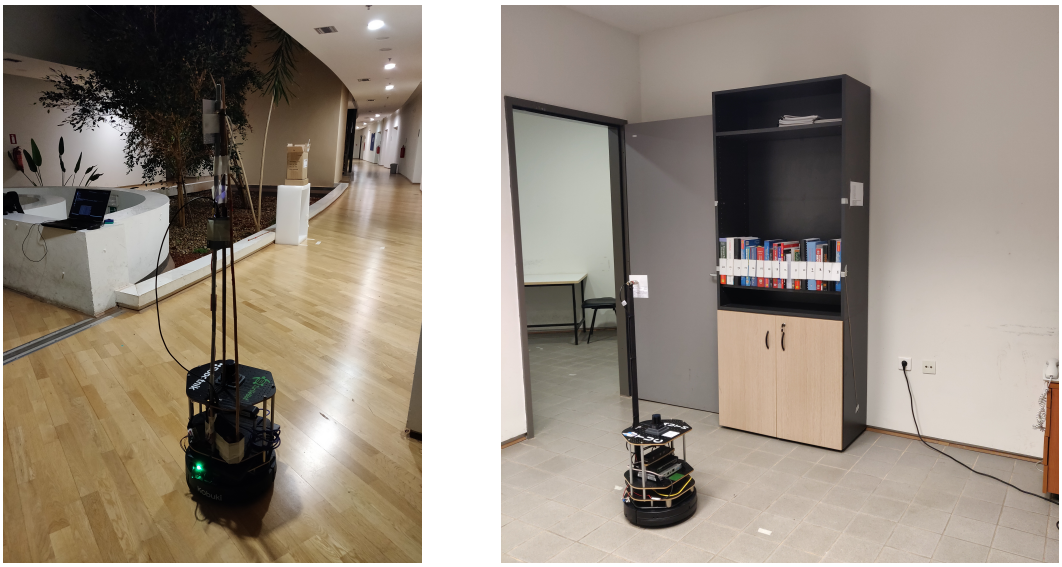


Figure 5.3: RMSE for various locations of the anchor tags under strong multipath.

Figure 5.4 contains photos from the environments where the experiments were conducted.

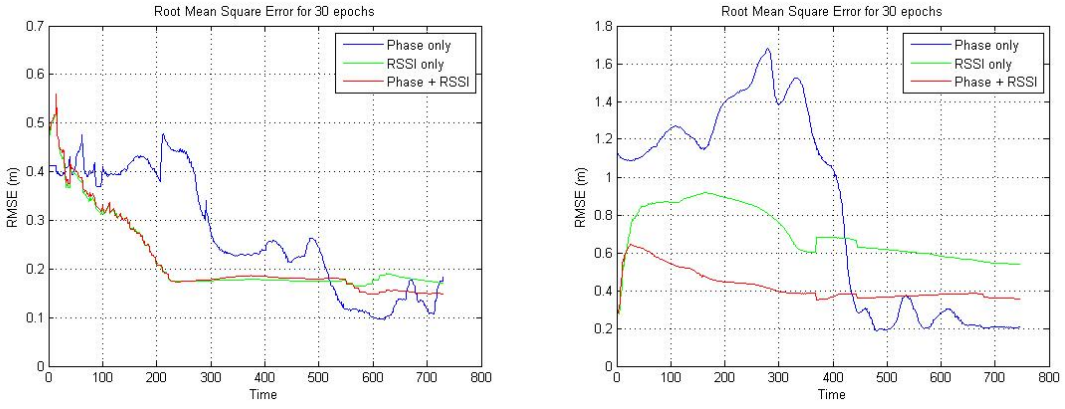


(a) Light multipath environment. (b) Strong multipath environment.

Figure 5.4: The environments of the experiments.

5.2.1 Localization in the 3D space

Figure 5.5 shows the results of the algorithm implemented in strong and light multipath environments (see figure 5.4). All cases are about localization in the 3D space and the search area's dimensions are 2m x 3m x 2m. An anchor tag is used for the estimation of the RSSI's distribution parameters (k, θ) located close to the unknown tags.



(a) Accuracy under light multipath. (b) Accuracy under strong multipath.

Figure 5.5: Comparison of localization accuracies - 3D.

Under light multipath, the noise in the RSSI observations is not much and the Gamma distribution is an accurate approximation for them. Actually, the RMSE of the algorithm which uses only the phase (10-20 cm) is about the same with the one which uses only the RSSI. Finally, their combination does not further improve the accuracy, but neither worsens it.

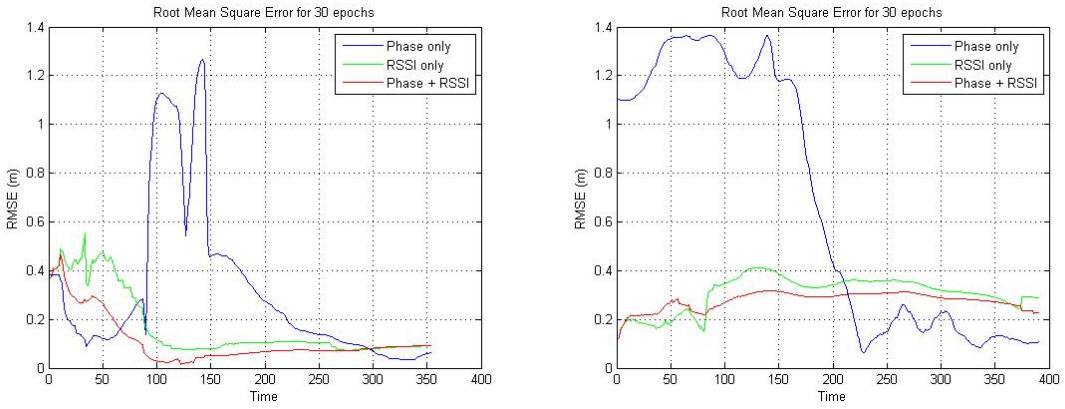
Under strong multipath, the proposed model (unimodal squared Gamma distribution for the RSSI observations which are multimodal) is not good, so the accuracy of the RSSI-based algorithm is not either. It's also worth noting that the accuracy of the phase-based algorithm remains about the same compared to the light multipath conditions. Finally, the RMSE of the combination is lower compared to the case in which only the RSSI is used, as it is affected by the phase.

Moreover, the accuracy when only phase is used is very low at the beginning of the algorithm, and it takes time until it is increased. For that time period, the combination of RSSI and phase is much better, since it produces the greatest accuracy achieved.

5.2.2 Localization in the 2D space

The results from the same algorithm implemented in 2D space (search area's dimensions equal with 2m x 3m) for both light and strong multipath (see figure 5.4) are shown in figure 5.6. Because of the less dimensions (i.e. variables inside the particles) to be estimated, the achieved accuracy is higher. As previously, an anchor tag is used for the estimation of the RSSI's distribution parameters (k, θ) and it is located close to the unknown tag.

The results are similar to the implementation in 3D space, but with each case having lower RMSE from its corresponding 3D one. Again, the lighter that the multipath is, the more accurate that the squared Gamma approximation for the RSSI's distribution is. Moreover, the RMSE of the RSSI-phase combination is lower than the only-RSSI case but not essentially lower than the only-phase situation.



(a) Accuracy under light multipath. (b) Accuracy under strong multipath.

Figure 5.6: Comparison of localization accuracies - 2D.

It can be easily understood that the filter's accuracy (when both RSSI and phase are used for the weight update) is mainly affected by the accuracy on the RSSI's model and not on the phase's. This can be reasoned because of the multimodality of the phase in contrast to the unimodality of the RSSI. The PDF of the phase has many peaks in concentric spheres with a radius difference $\lambda/2$, and the weight update of a particle is related to the nearest peak. On the contrary, the RSSI model has just one peak, common for all particles, and the weight update is related to their distance from it. An error on the RSSI will move the total estimate to the wrong direction, with the phase just making slight corrections to it (and so is reasoned the lower RMSE in the strong multipath situation, compared to the only-RSSI approach).

However, the combination of RSSI along with the phase offers a reliable estimation algorithm. It can be used in both light and strong multipath and it achieves an RMSE not greater than the corresponding algorithm which uses only the RSSI. Moreover, it does not need time until it converges, since the RMSE does not vary significantly while time passes, but it remains about the same.

Chapter 6

Conclusions

This work summarizes the most popular strategies for indoor navigation. More emphasis was given to the localization and SLAM algorithms, as yet they are open for research problems. Two of the most popular technologies were mentioned; the visual one with many packages implemented and ready-to-use, and the RFID one which is a popular research topic.

Moreover, an approach to combine RSSI and phase inside a particle filter was proposed, so that a further increase on the accuracy is achieved. RSSI is modeled by a scaled squared variable which follows the Gamma distribution. Additionally, maximum likelihood estimation is used to estimate the parameters of the distribution. However, in cases of strong multipath the localization error is increased as the specific distribution does not approximate the model of the measurements as good as under light multipath conditions.

Relatively to the numerical results, it is obvious that the smaller the distance between the anchor tag and the unknown is, the better the estimation of the parameters is too, especially when that distance is less than half the wavelength and the measurements of the tags are correlated. When only the phase is used for the weight update of the particles, the localization error is initially increased, but it is reduced while time passes. On the other hand, when only the RSSI is used the localization error is higher than before, but it does not increase at the beginning either. Finally, when the two measurements are combined, the localization error is less compared to when only the RSSI is used and it does not increase significantly at the beginning, making that solution suitable for all environments without the need for waiting until the algorithm converges. Of course, in environments with rich multipath the estimation error will be higher compared to light or no multipath.

Future work may include a further analysis for accurate RSSI models under strong multipath conditions, by implementing the Expectation-Maximization algorithm modified to use complex data. Also, the phase is affected by the power of the received signal (RSSI), thus a study on that topic would be interesting too (with such a study to prerequisite a good modeling of the RSSI, first). Finally, more complex resampling strategies for the phase model should be also examined, in order that its unimodality is preserved instead of approximated by a Gaussian.

A Conversion of multistatic RSSI model to monostatic

According to previous work[7], the power of the received signal at time t equals with:

$$y = P_{Tx} n_T G_{Tx} G_T^2 G_{Rx} L_{Tx}^2 L_{Rx}^2 |h_{Tx}|^2 |h_{Rx}|^2$$

with

$$L_{Tx} = \begin{cases} \left(\frac{\lambda}{4\pi||x_{Tx}-x_T||_2}\right), & if||x_{Tx}-x_T||_2 < \frac{4\pi h_{Tx} h_T}{\lambda} \\ \left(\frac{h_{Tx} h_T}{||x_{Tx}-x_T||_2^2}\right)^2, & if||x_{Tx}-x_T||_2 \geq \frac{4\pi h_{Tx} h_T}{\lambda} \end{cases}$$

$$L_{Rx} = \begin{cases} \left(\frac{\lambda}{4\pi||x_{Rx}-x_T||_2}\right), & if||x_{Rx}-x_T||_2 < \frac{4\pi h_{Rx} h_T}{\lambda} \\ \left(\frac{h_{Rx} h_T}{||x_{Rx}-x_T||_2^2}\right)^2, & if||x_{Rx}-x_T||_2 \geq \frac{4\pi h_{Rx} h_T}{\lambda} \end{cases}$$

P_{Tx} refers to the transmission power and n_T to the tag scattering efficiency. Also, $G_{Tx/Rx/T}$ is the gain of the transmitter's/receiver's/tag's antennas respectively. Similarly, $x_{Tx/Rx/T}$ indicates the transmitter's/receiver's/tag's locations and $h_{Tx/Rx/T}$ their heights respectively. Finally, λ is the wavelength of the radiation and $|h_{Tx/Rx}|$ are independent Nakagami random variables modeling the transmitter-to-tag and tag-to-reader fading coefficients (thus, $|h_{Tx/Rx}|^2$ are independent Gamma variables).

This is a multistatic model, which means that the location of the transmitter (of the unmodulated signal) and the reader (of the modulated by the tag signal) is different. For the purposes of this thesis it has to be converted to a monostatic one, because the same antenna is used both as the transmitter's and the reader's antenna. To keep things simple the 2-ray model is used, which is an accurate enough model for environments with light multipath (not many surfaces exist for the signal to be reflected). However, for environments with strong multipath this model is significantly inaccurate compared to the actual measurements.

In the case of a monostatic model, the same device both transmits a signal and reads the tag's response. That simplifies the above formula as follows:

- $G_{Tx} = G_{Rx}$ since the transmitting and receiving antennas are the same one.
- $||x_{Tx} - x_T||_2 = ||x_{Rx} - x_T||_2$ since the transmitting and receiving antennas are the same one.
- $|h_{Tx}|^2 = |h_{Rx}|^2$ since the reader-to-tag and tag-to-reader channels are the same one.

Thus, in a monostatic model the power of the received signal equals with:

$$y = P_{Tx} n_T G_R^2 G_T^2 L^2 |h|^4 = \begin{cases} P_{Tx} n_T G_R^2 G_T^2 \left(\frac{\lambda}{4\pi||x_R-x_T||_2}\right)^4 |h|^4, & if||x_R-x_T||_2 \leq \frac{4\pi h_R h_T}{\lambda} \\ P_{Tx} n_T G_R^2 G_T^2 \left(\frac{h_R h_T}{||x_R-x_T||_2^2}\right)^4 |h|^4, & if||x_R-x_T||_2 > \frac{4\pi h_R h_T}{\lambda} \end{cases}$$

For the monostatic formula's symbols explained also refer to 4.1.2.

B Importance sampling in particle filters

Importance sampling [8] [17] (or resampling) is used to transform the distribution of the particle set into another one. The new set of particles will be distributed according to the posterior probability, and particles will not be spent in areas with low probability density.

However, repetitive resampling induces loss to the diversity of the particles (reduction to the variance of the samples), which may lead to particle deprivation. That's because of the random nature of resampling, due to which some particles may be excluded (not resampled) and consequently, be lost. Many different resampling strategies have been suggested [17] with each one having both advantages and disadvantages. Two easy-to-implement techniques that improve significantly the performance of importance sampling are analyzed below:

- **Low Variance Sampling (LVS):** This algorithm is about the method of the resampling process (i.e. the way in which M new particles are sampled from the existing ones, depending on their weights). Their selection does not take place independently of each other, but it involves a sequential stochastic process. Yet, the probability for selection remains proportional to each particle's weight.

Algorithm Low Variance Sampling (LVS)

Required:

M, the number of the particles

$\bar{X}_t = \{\bar{x}_t^{[1]}, ..., \bar{x}_t^{[M]}\}$, the set of particles which will be sampled.

$\bar{W}_t = \{\bar{w}_t^{[1]}, ..., \bar{w}_t^{[M]}\}$, the normalized weights of the particles in $\bar{X}_t$.

Returned:

$X_t = \{x_t^{[1]}, ..., x_t^{[M]}\}$, the set of the sampled particles.

Pseudocode:

1: $X_t \leftarrow \emptyset$
2: $r \leftarrow rand\left(0, \frac{1}{M}\right)$
3: $c \leftarrow \bar{w}_t^{[1]}$
4: $i \leftarrow 1$
5: **for** m← 1:M **do**
6: $U \leftarrow r + \frac{m-1}{M}$
7: **while** $U > c$ **do**
8: $i \leftarrow i + 1$
9: $c \leftarrow c + \bar{w}_t^{[i]}$
10: **end while**
11: $X_t \leftarrow X_t + \{\bar{x}_t^{[i]}\}$
12: **end for**

LVS covers the samples in a more systematic way, compared to choosing some random numbers and getting the particles which correspond to them (independently from each other). Actually, if the weights of a sample set are all

equal, then the returned by the algorithm set will be the same as the input one. Moreover, the complexity of resampling using the LVS algorithm is linear with respect to the number of particles ($O(M)$) in contrast with other implementations whose complexity is greater ($O(M \log M)$).

- **Reduction of the resampling frequency:** Too frequent resampling may lead to loss of diversity among the particles, while rare resampling wastes too many particles in low probability regions. The estimation of the right resampling frequency is important and the variance of the weights is a good enough criterion for that. Resampling will only take place when $N_{eff} = \frac{1}{\sum_{m=1}^{M}\left(\bar{w}^{[m]}\right)^2} < N_{thresh}$ with N_{eff} being the Effective Sample Size and $\bar{w}^{[1,2,\ldots,M]}$ the normalized weights of the particles.

  Regarding the weight updates at each time step, they will be set equal to 1 after each resampling, or they will be multiplied with the respective weights of the previous time step when no resampling takes place:

$$w_t^{[m]} = \begin{cases} 1, & after\ resampling \\ w_{t-1}^{[m]} p\left(z_t | x_t^{[m]}\right), & after\ no\ resampling \end{cases}$$

  Since $0 \leq \bar{w}^{[m]} \leq 1$ and $\sum_{m=1}^{M} \bar{w}^{[m]} = 1$, it is induced that $1 \leq N_{eff} \leq M$. A typical value for the fixed threshold is $N_{thresh} = \frac{M}{2}$. The smaller the threshold is, the longer it takes for the estimation error to be reduced. The higher the threshold is, the greater the danger for particle deprivation is.

C PDF of a scaled squared Gamma variable

Let $Z = aY = aX^2$ with $X \sim \Gamma(k, \theta)$ (=Gamma distribution with shape k, scale θ) and $a, Y, X \geq 0$. It is induced that $f_X(x) = \frac{x^{k-1}e^{-\frac{x}{\theta}}}{\theta^k \Gamma(k)}$.

$$F_Y(y) = P(Y \leq y) = P(X^2 \leq y) = P(|X| \leq \sqrt{y}) = P(X \leq \sqrt{y}) = F_X(\sqrt{y}) \implies$$

$$\implies f_Y(y) = \frac{dF_Y(y)}{dy} = \frac{f_X(\sqrt{y})}{2\sqrt{y}}$$

$$f_Z(z) = \frac{1}{a} f_Y\left(\frac{z}{a}\right) = \frac{f_X(\sqrt{\frac{z}{a}})}{2a\sqrt{z}a} = \frac{1}{2\sqrt{az}} \frac{\sqrt{\frac{z}{a}}^{k-1} e^{-\frac{\sqrt{\frac{z}{a}}}{\theta}}}{\theta^k \Gamma(k)} = \frac{1}{2\sqrt{az}} \frac{(\frac{z}{a})^{\frac{k-1}{2}} e^{-\frac{1}{\theta}\sqrt{\frac{z}{a}}}}{\theta^k \Gamma(k)}$$

D Estimation of (k, θ) parameters for a scaled squared Gamma variable

Let $X = aZ^2$, with $a \in \mathbb{R}$ (known variable) and $Z \sim \Gamma(k, \theta)$. The problem is to determine the values of k, θ given a set of N independent measurements $\{x_1, x_2, ..., x_N\}$ for the variable X and the respective $\{a_1, a_2, ..., a_N\}$ variables. Maximum Likelihood (ML) estimation will be used to solve this problem.

The likelihood function is:

$$L(k,\theta) = \prod_{i=1}^{N} f(x_i, a_i; k, \theta) = \prod_{i=1}^{N} \frac{1}{2\theta^k \Gamma(k)\sqrt{a_i x_i}} \left(\frac{x_i}{a_i}\right)^{\frac{k-1}{2}} \left(e^{-\frac{1}{\theta}\sqrt{\frac{x_i}{a_i}}}\right)$$

The log-likelihood function is:

$$\ell(k,\theta) = \ln(L(k,\theta)) = \ln\left(\prod_{i=1}^{N}\left(\frac{x_i}{a_i}\right)^{\frac{k-1}{2}}\right) + \ln\left(\prod_{i=1}^{N} e^{-\frac{1}{\theta}\sqrt{\frac{x_i}{a_i}}}\right) - \ln\left(\prod_{i=1}^{N} 2\sqrt{a_i x_i}\theta^k \Gamma(k)\right) =$$

$$= \frac{k-1}{2}\sum_{i=1}^{N}\ln\left(\frac{x_i}{a_i}\right) - \frac{1}{\theta}\sum_{i=1}^{N}\sqrt{\frac{x_i}{a_i}} - \frac{1}{2}\sum_{i=1}^{N}\ln(a_i x_i) - N\ln(2) - Nk\ln(\theta) - N\ln(\Gamma(k))$$

The log-likelihood function has to be maximized with respect to θ, by setting its partial derivative equal to zero.

Let $A = \frac{k-1}{2}\sum_{i=1}^{N}\ln\left(\frac{x_i}{a_i}\right) - \frac{1}{2}\sum_{i=1}^{N}\ln(a_i x_i) - N\ln(2) - N\ln(\Gamma(k))$ and $B = \sum_{i=1}^{N}\sqrt{\frac{x_i}{a_i}}$.

Then, $\ell(k,\theta) = \frac{-B}{\theta} - Nk\ln(\theta) + A$.

$$\frac{\partial \ell}{\partial \theta} = \frac{B}{\theta^2} - \frac{Nk}{\theta} = \frac{B - Nk\theta}{\theta^2}$$

As stated above, to compute the maximum likelihood estimator of the θ parameter it is required that:

$$\frac{\partial \ell}{\partial \theta} = 0 \Leftrightarrow \hat{\theta} = \frac{B}{Nk} \Leftrightarrow \hat{\theta} = \frac{1}{Nk}\sum_{i=1}^{N}\sqrt{\frac{x_i}{a_i}}$$

By replacing θ with its ML estimator in ℓ, it is induced that:

$$\ell(k) = \frac{k-1}{2}\sum_{i=1}^{N}\ln\left(\frac{x_i}{a_i}\right) - \sum_{i=1}^{N}\ln\left(2\sqrt{a_i x_i}\right) - N\ln\left(\Gamma(k)\right) - Nk - Nk\ln\left(\sum_{i=1}^{N}\sqrt{\frac{x_i}{a_i}}\right) + Nk\ln(Nk)$$

The partial derivative of ℓ with respect to k is:

$$\frac{\partial \ell}{\partial k} = N\ln(N) + N\ln(k) - N\psi(k) + \sum_{i=1}^{N}\ln\left(\sqrt{\frac{x_i}{a_i}}\right) - N\ln\left(\sum_{i=1}^{N}\sqrt{\frac{x_i}{a_i}}\right)$$

The ML estimator of the k parameter will be computed the same way as above, by setting the partial derivative equal to zero:

$$\frac{\partial \ell}{\partial k} = 0 \Leftrightarrow \ln(k) - \psi(k) = \ln\left(\frac{1}{N}\sum_{i=1}^{N}\sqrt{\frac{x_i}{a_i}}\right) - \frac{1}{N}\sum_{i=1}^{N}\ln\left(\sqrt{\frac{x_i}{a_i}}\right)$$

There is no closed-form solution to this equation, so an approximation will be used instead. With the proof given at the appendix E, it can be shown that: $\ln(k) - \psi(k) \approx \frac{1}{2k}\left(1 + \frac{1}{6k}\right)$. Thus, if it is denoted $s = \ln\left(\frac{1}{N}\sum_{i=1}^{N}\sqrt{\frac{x_i}{a_i}}\right) - \frac{1}{N}\sum_{i=1}^{N}\ln\left(\sqrt{\frac{x_i}{a_i}}\right)$, the ML estimator for parameter k is:

$$\frac{1}{2\hat{k}}\left(1 + \frac{1}{6\hat{k}}\right) \approx s \Leftrightarrow \hat{k} \approx \frac{3 + \sqrt{9 + 12s}}{12s}$$

The accuracy of the ML estimators in contrast to the sample size can be seen in figure 1.

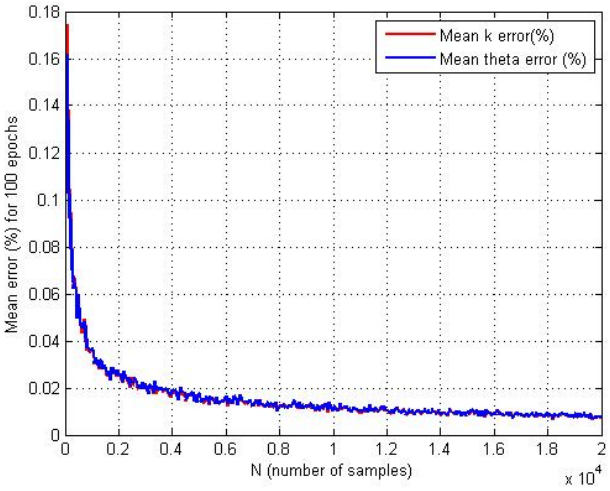


Figure 1: Accuracy of the $(\hat{k}, \hat{\theta})$ ML estimators.

E Stirling approximation

The Stirling Series formula states[18] that: $\ln(n!) = n\ln(n) - n + \frac{1}{2}\ln(n) + \ln(\sqrt{2\pi}) + \sum_{k=1}^{\infty} \frac{B_{2k}}{2k(2k-1)n^{2k-1}}$ where B_m is the $m-th$ Bernoulli number. For greater accuracy, the 2^{nd} order approximation will be used, that is:

$$\ln(n!) \approx n\ln(n) - n + \frac{1}{2}\ln(n) + \ln(\sqrt{2\pi}) + \frac{1}{12n} \implies$$

$$\implies \ln(n\Gamma(n)) \approx n\ln(n) - n + \frac{1}{2}\ln(n) + \ln(\sqrt{2\pi}) + \frac{1}{12n} \xrightarrow{\partial/\partial n}$$

$$\implies \psi(n) = \ln(n) - \frac{1}{2n} - \frac{1}{12n^2} \implies \ln(n) - \psi(n) = \frac{1}{2n}\left(1 + \frac{1}{6n}\right)$$

Accuracy of the 2^{nd} order approximation: The approximation of $\ln(n!)$ using the 2^{nd} order Stirling Series is extremely precise[19]. On the table below, the difference between the Digamma function ($\psi(n)$) and its approximation $\left(\ln(n) - \frac{1}{2n}\left(1 + \frac{1}{6n}\right)\right)$ can be seen (in % percentage of the Digamma's value), for various values of n.

n	0.5	0.8	1	3	5	10	20
Difference (%)	3.21	1.38	1.06	0.01	$8.69 * 10^{-4}$	$3.68 * 10^{-5}$	$1.75 * 10^{-6}$

Thus, for any $n > 1$ the error due to the approximation will be less than 1%.

F Localization approach with the use of a motion model

Algorithm 8 Localization of a moving robot using anchor tags

Set M (number of particles), T (time window), d_{max}^{reader} (reader's max distance to detect a tag), N_{thresh}
Set $X_{min}^{a}, X_{max}^{a}, Y_{min}^{a}, Y_{max}^{a}, Z_{min}^{a}, Z_{max}^{a}$ (search area dimensions)
Set $\{g^{[1]}, g^{[2]}, ..., g^{[G]}\}$ (the positions of the anchor tags)
for m$\leftarrow$ 1:M **do**
    $x^{[m]} \sim U\left[X_{min}^{a}, X_{max}^{a}\right], y^{[m]} \sim U\left[Y_{min}^{a}, Y_{max}^{a}\right], z^{[m]} \sim U\left[Z_{min}^{a}, Z_{max}^{a}\right]$
    $\phi^{[m]} \sim U\left[\phi_{min}^{a}, \phi_{max}^{a}\right], \theta_{i}^{[m]} \sim U\left[0, 2\pi\right) \forall i = 1, 2, ..., G$
    $X_{0}^{[m]} \leftarrow \left[x^{[m]}\ y^{[m]}\ z^{[m]}\ \phi^{[m]}\ \theta_{1}^{[m]}\ \theta_{2}^{[m]}\ ...\ \theta_{G}^{[m]}\right], w^{[m]} \leftarrow 1$
end for
for t$\leftarrow$ 1:T **do**
    Collect the RSSI $\{y_{t}^{[1]}, y_{t}^{[2]}, ..., y_{t}^{[K]}\}$ & the phase measurements $\{\phi_{t}^{[1]}, \phi_{t}^{[2]}, ..., \phi_{t}^{[K]}\}$, $K \leq G$.
    Collect the robot's odometry dX_t.
    for m$\leftarrow$ 1:M **do**
        $X_{t}^{[m]} \sim N\left(X_{t-1}^{[m]} + dX_t, \sigma_{odometry}^{2}\right)$
        for j$\leftarrow$ 1:G **do**
            $W_{RSSI} \leftarrow p\left(y_{t}^{[j]} | X_{t}^{[m]}\right) \equiv \frac{1}{2\theta^{k}\Gamma(k)\left(a_{t}^{[j,m]}\right)^{k/2}} \left(y_{t}^{[j]}\right)^{\frac{k-2}{2}} \left(e^{-\frac{1}{\theta\sqrt{a_{t}^{[j,m]}}}\sqrt{y_{t}^{[j]}}}\right)$
            $W_{PHASE} \leftarrow p(\Delta_{t}^{[j,m]} | X_{t}^{[m]}) \equiv N(\Delta_{t}^{[j,m]}; 0, \sigma_{phase}^{2})$
            $w^{[m]} \leftarrow w^{[m]} W_{RSSI} W_{PHASE}$
        end for
    end for
    $w^{[1:M]} \leftarrow \frac{w^{[1:M]}}{\sum_{m=1}^{M} w^{[m]}}$ ▷ Normalization of weights
    $N_{eff} \leftarrow \frac{1}{\sum_{m=1}^{M}\left(w^{[m]}\right)^{2}}$
    if $N_{eff} < N_{thresh}$ **then** ▷ Resampling if needed
        $X_{LVS}^{[1:M]} \leftarrow LVS\left(X_{t}^{[1:M]}, w^{[1:M]}\right)$
        $w^{[1:M]} \leftarrow \frac{1}{M}$
    end if
    $\hat{X}_{est}^{[t]} = \sum_{m=1}^{M} w^{[m]} X_{t}^{[m]}$ ▷ Position estimation for time t
end for
return $\hat{X}_{est}$

References

[1] S. F. R. Alves, J. M. Rosario, H. F. Filho, L. K. A. Rincon, and R. A. T. Yamasaki, "Conceptual Bases of Robot Navigation Modeling, Control and Applications," in *Advances in Robot Navigation*, A. Barrera, Ed. IntechOpen, 2011.

[2] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni, "Path Planning and Trajectory Planning Algorithms: A General Overview," *Mechanisms and Machine Science*, vol. 29, pp. 3–27, Mar. 2015.

[3] T. Nick, S. Cordes, J. Götze, and W. John, "Camera-assisted localization of passive RFID labels," in *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1-8, Nov. 2012, Sydney, NSW, Australia.

[4] F. Martinelli, "Simultaneous Localization and Mapping Using the Phase of Passive UHF-RFID Signals," *Journal of Intelligent & Robotic Systems*, vol. 94, pp. 1–15, Jul. 2018.

[5] F. Martinelli, "A Robot Localization System Combining RSSI and Phase Shift in UHF-RFID Signals," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 5, pp. 1782–1796, Sep. 2015.

[6] E. Giannelos, "Intelligent Wireless Networks and Robots for Low-Cost Battery-Less Sensing and Localization," Ph.D. dissertation under preparation, Technical University of Crete, Chania, Greece, advisor: A. Bletsas.

[7] M. Vestakis, P. N. Alevizos, G. Vougioukas, and A. Bletsas, "Multistatic Narrowband Localization in Backscatter Sensor Networks," in *IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 1-5, Jun. 2018, Kalamata, Greece.

[8] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA: The MIT Press, 2006.

[9] D. Fox, "KLD-Sampling: Adaptive Particle Filters," in *Proceedings of the 14th International Conference on Neural Information Processing Systems (NIPS): Natural and Synthetic*, pp.713-720, Jan. 2001, Vancouver, British Columbia, Canada.

[10] A. Burchardt, T. Laue, and T. Röfer, "Optimizing Particle Filter Parameters for Self-localization," in *RoboCup 2010: Robot Soccer World Cup XIV*, J. Ruiz-del

Solar, E. Chown, and P. G. Plöger, Eds. Berlin, Heidelberg: Springer, 2011, pp. 145–156.

[11] G. Grisetti, C. Stachniss, and W. Burgard, "Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, Feb. 2007.

[12] A. Nayegandhi, "LiDAR technology overview," *US Geological Survey*, 2007.

[13] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1-4, May 2011, Shanghai, China.

[14] B. Ruf, S. Monka, M. Kollmann, and M. Grinberg, "Real-time on-board obstacle avoidance for UAVs based on embedded stereo vision," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLII-1, pp. 363–370, Sep. 2018.

[15] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-Time Loop Closure in 2D LIDAR SLAM," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1271-1278, May 2016, Stockholm, Sweden.

[16] R. Want, "An introduction to RFID technology," *IEEE Pervasive Computing*, vol. 5, no. 1, pp. 25–33, Jan.-Mar. 2006.

[17] T. Li, M. Bolic, and P. M. Djuric, "Resampling Methods for Particle Filtering: Classification, implementation, and strategies," *IEEE Signal Processing Magazine*, vol. 32, no. 3, pp. 70–86, May 2015.

[18] J. Gélinas, "Original proofs of Stirling's series for log(n!)," Jan. 2017.

[19] W. Feller, *An introduction to probability theory and its applications, Vol 2.* John Wiley & Sons, 2008.