


University of Montpellier - LIRMM
Robotics Master Project Report 2021/2022

# Particle filter localization: Off-line performance evaluation

**Authors** :

- Abderahmane BENALI
- Zakaria BOUZIT

**Supervisors** :

- Mr. Didier CRESTANI
- Mr. Philippe LAMBERT

**Reviewer** :

- Mr. Tho DANG-HUU

# Contents

# List of Figures

**ABSTRACT**

Mobile robot global localization is the problem of determining a robot's pose in an environment by using sensor data, when the initial position is unknown.

In this context, Monte Carlo Localization is one of the most popular approaches, and represents a good trade-off between robustness and accuracy. The basic underlying principle of this family of approaches is using Particle filter for tracking a probability distribution of the possible robot poses.

In this work, we focus on analyzing Particle filter parameters and performances qualification in terms of uncertainties and adaptability to a known environment.

***Keywords***— Mobile robotics, Global localization, Particle filter

# 1 Problem formulation

## 1.1 Mobile robotics and localization

Mobile robots have remarkable achievements in industrial manufacturing, medical science, agriculture, space research, etc. Where the main task of a mobile robot is the ability to navigate autonomously and perform successful navigation. In this kind of mission, the robot passes through different phases such as perception, localization, cognition, and motion control.

In the perception phase, the robot extracts meaningful data by interpreting its sensors information. In the localization phase, the robot estimates its current location in its environment using information from internal and external sensors. In the cognition phase, the robot plans the necessary steps to reach the target. Then, the motion control phase lets the robot achieve its desired trajectory by modifying its actuators outputs. The rest of this paper focuses on the localization problem, since knowing regularly where we are in our environment is one of the most fundamental competencies required by an autonomous robot.

### 1.1.1 Localization methods

Mobile robot localization is the process of tracking the pose of the robot relatively to its environment. There are two types of localization, local and global. Local techniques aim to compensate odometric errors during robot navigation. They require that the initial location of the robot is approximately known and they typically cannot recover if they lose track of the robot's position. Global techniques can localize a robot without any prior knowledge about its position, i.e., they can handle the loss of the robot pose. [2].

Particle filter is one of the global localization techniques that adopt probabilistic techniques which integrates imperfect models and imperfect sensors through probabilistic laws [3]. Which make this method suitable to constraints of noisy and incomplete sensors measurement provided by the robot, and the complexity of mission environment.

## 1.2 Work context and objectives

The problem addressed consists of localizing a mobile robot **Pioneer-P3Dx** equipped with lasers, ultrasounds, and odometry sensors, in a known environment during an autonomous mission : navigate from a start point to a given final point. And during this mission we need to locate our robot correctly with known and acceptable performances.

This work is a continuation of the work done last year (2020-2021) in the framework of the Master robotics of the Faculty of Sciences of Montpellier.[4]

We had a first implementation of the simulation with the robot control and particles motion simulation and all the sensors models in Matlab. So our work was mainly focused on the Localization method (particle filter), study cases, and performances qualification (in simulation) in terms of uncertainties and adaptability to our environment. With the goal to identify the environment areas allowing to use an efficient particle filter localization and its associated parameters configuration. [1] [4]

### 1.2.1 Mission Environment

The robot should accomplish the mission in indoor environment, which make the localization a challenging task. First, because human buildings are composed of a set of rooms and corridors strongly structured and symmetrical which causes the phenomenon of perceptual aliasing. Second, we don't have, as outside with the GPS, an omniscient system, capable of localizing almost at any moment, with a known precision, a mobile system.[1]

Figure 1 presents an overall view of the mission environment, we distinguish 3 zones. The experimentation hall where the robot is initially located, a first corridor CR1, and a second corridor CR2. The performance was established within an experimental area (corridor 2) of about 30 meters long with optimal lighting and equipped with many QR-codes used to qualify the performances of the localization method. [1].

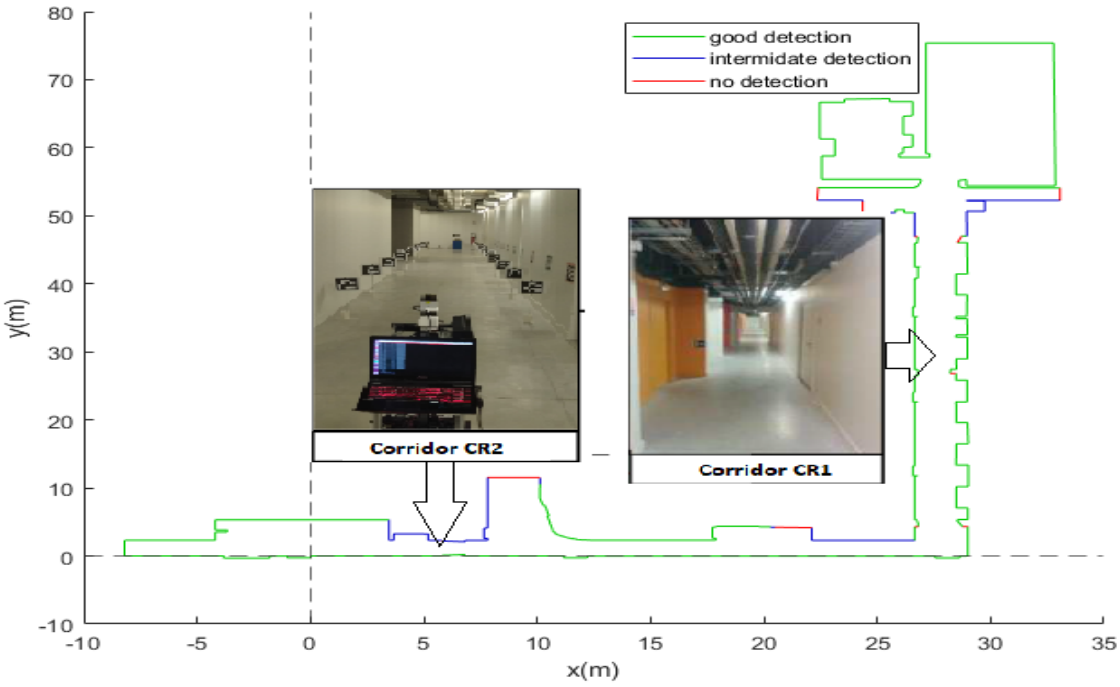



**Figure 1:** Overall view of the mission environment [1]

### 1.2.2 The mobile platform

We use a non-holonomic robot Pioneer-P3Dx to realize the mission. Basically composed of a mobile differential base with 2 driving wheels, an idler wheel, ultrasounds sensors, banks and bumpers.

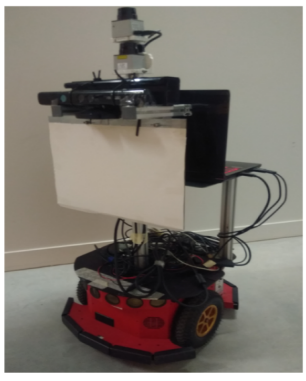


**Figure 2:** The mobile platform [1]

It has been added to this base a pair of laser sensors mounted head to tail on top of the platform, and a table supporting a laptop to supervise the

mission. [1]

**Ultrasounds sensors**

We have 16 ultrasounds sensors, with their repartition illustrated in figure 3. The US sensors calculate the distance by measuring the time of flight of an acoustic wave, the US range is from $0.1m$ to $4m$.

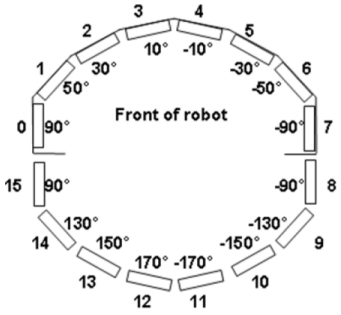



**Figure 3:** US sensors in our platform

**Laser sensors**

We have two laser sensors that cover 240°each, the first one is oriented to the front of the robot and the second to the back. Each laser sends 683 laser ray in different directions and measures the distance from obstacles in that direction. In total, both lasers cover 360° of the robot in 1022 points (without counting the repeated points). Note that laser measurement are not precise in some parts of our environment, with no detection in some parts, as shown in figure1.

## 1.3 Conclusion

This report will be organized into 4 parts. It starts with the problem formulation and Work context, then the Particle filter localization is presented, its principle and theory, then we talk about the studied parameters and cases to end with performances evaluation in term of convergence, validation, and adaptability to our environment.

# 2 Particle filter

In this section we introduce the particle filter method, and its steps.

## 2.1 Particle filter for localization

A Particle filter is a sequential Monte Carlo method used to solve filtering problems (state estimation)[5], in or case we use the particle filter to estimate a robot's pose (position and orientation) in an indoor environment 1 and sense its movements. For that, we suppose that we have a swarm of points called particles that have the same state vector representation as the robot state vector $\chi = [x, y, \theta]^T$, Where $x$ and $y$ representing the robot position in the environment and $\theta$ the orientation of the robot in the environment frame 1.

Each particle represents a possible state $\chi_i$ for the robot (hypothesis of where the robot is) with a probability of the robot being present in that pose (weight $w_i = P(\chi_i|\chi)$). Considering all the particles present in the environment we select the best candidates and estimate the robot pose based on these particles. [5] [6].

## 2.2 Particle filter Algorithm

The Particle filter algorithm proceeds as follows : [7]

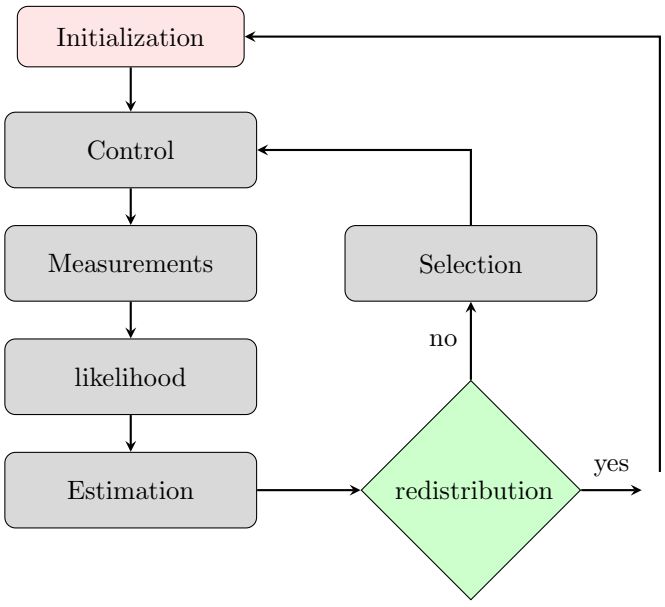



**Figure 4:** Particle filter algorithm

Here we will detail each step of the algorithm.

### 2.2.1 Initialization

$N$ particles are randomly generated in the environment, each particle is a $[3 \times 1]$ state vector representing the robot pose. The particles are uniformly distributed in the map. [4].

### 2.2.2 Control

First, a PID controller is used to control the robot. The goal is to follow a set of target points representing a given trajectory. Then, the same control law is applied for all the sets of particles, this will result in having an N different version of the robot state vector.

### 2.2.3 Measurements

For each particle "$i$", the observation vector is predicted using a simulation model of the used sensor and compared to the actual observation vector of the robot.
For both sensors available on our robot, the polar coordinate are used to represent the observation vector, with an orientation $\hat{\theta}[i, j]$ (for the sensor "$j$" orientation in case of ultrasonic sensor or the ray "$j$" orientation in case of laser sensor) and a measured distance $\hat{r}_{particle}[i, j]$ between the particle and the obstacle in that direction.

### 2.2.4 Likelihood

The measured distance is compared for each particle "$i$" with the actual measured distance of the robot. For this many supposition are done. First, a measurement $\hat{r}_{particle}[i]$ of a particle is in some way related to the state we wish to estimate [7][8]. Also, the measurement of the robot state is not precise (noisy measurement).

The distribution of a particle is defined by the conditional probability of the measurement $\hat{r}_{particle}[i]$ given a robot measurement $\hat{r}_{robot}$. In our case, the weight of the particle is given by a Gaussian law. [4].

- For each particle "$i$", the associated weight "$w[i]$" is:

$$\delta_i = \hat{r}_{robot} - \hat{r}_{particle}[i] \tag{1}$$

$$\sigma[i] = \sqrt{var(\delta_i)} \tag{2}$$

$$E_i = (\delta_i - mean(\delta_i))/\sigma[i] \tag{3}$$

$$w[i] = \frac{1}{\sqrt{2.\pi}}.e^{-0.5.E_i^T.E_i} \tag{4}$$

Where $\hat{r}_{robot}$ and $\hat{r}_{particle}[i]$ are vectors with $j$ distances, $\delta_i$ is the difference between particle $i$ measurements and the robot measurements, $\sigma[i]$ is the standard deviation, $w[i]$ is the weight associated to the particle "$i$".

### 2.2.5 Estimation

After calculating the weights of each particle, the robot pose is calculated based on all particles poses and weights : [9]

$$x_{estimated} = \sum_{i=1}^{N} \overline{w}(i) \ . \ x_{particle}(i) \tag{5}$$

$$y_{estimated} = \sum_{i=1}^{N} \overline{w}(i) \ . \ y_{particle}(i) \tag{6}$$

$$\theta_{estimated} = \sum_{i=1}^{N} \overline{w}(i) \ . \ \theta_{particle}(i) \tag{7}$$

With $\overline{w}(i)$ is the normalized weight ($\overline{w}(i) = \frac{w(i)}{\sum_{i=1}^{N} w_i}$)

Then we add a criterion to validate this pose, a pose is supposed valid if it is close enough to the real robot pose. Since such information is not available, another criterion is used to validate the pose that will be discussed in the next section.

### 2.2.6 Particle Selection

Select the particles that best explain the observation. There are many ways of doing this, the obvious one is simply to sort and select the top candidates (the particles with the biggest weights), but this would reduce the number of particles. One solution to this would be copying the best candidates until having the N particles again, the consequence of this is reducing the diversity and spread of the particle set.

Instead, we randomly sample from the samples biased by the importance's values. for this two methods are presented with different approaches. In both, there is a chance to pick a particle that didn't necessarily explain the observation well, but this may turn out to be a good choice because it may do a better job for the next observation. After picking all the particles, we randomly perturb them [4] this is equivalent to adding $Q$ in the extended Kalman filter [7].

### 2.2.7 Redistribution

If the weights are too concentrated, the particles are redistributed in all the environment. This step has two advantages, first, the particles may be concentrated around the wrong pose, then a redistribution of particles will fix this. Second, this step is important in the case of a kidnapping phenomenon, kidnapping phenomenon is a situation where a well-localized mobile robot is teleported to an arbitrary location without being told, this can be a result of a sensor failure thus, so the Particle filter should have the ability to recover the robot pose even after a localization failure occurs [10]. The redistribution criterion is discussed in the next section.

## 2.3 Conclusion

The particle filter is used in our work to estimate the pose of the robot using a simulation of the sensors model with $N$ particles having the same pose vector as the robot. After discussing all the particle filter steps in this section, we will focus on evaluating the particle filter parameters and the corresponding performances, in the next one.

# 3 Particle filter parameters evaluation

## 3.1 introduction

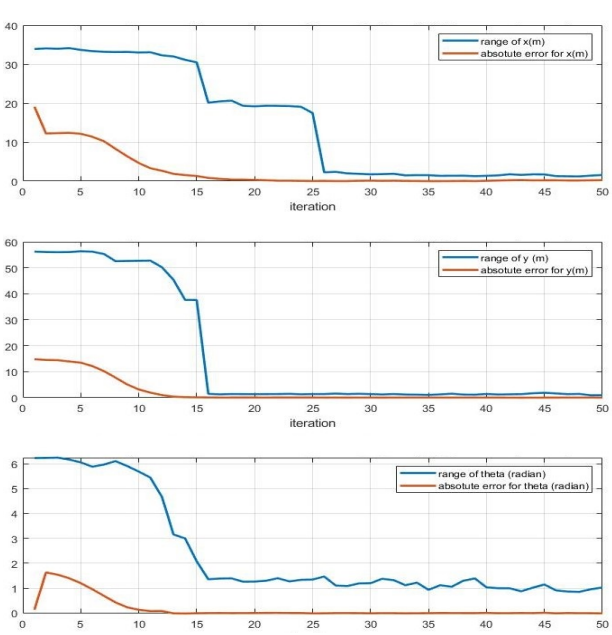
The particle filter performance is defined by the number of convergence time and the error between the estimated position and robot position(precision).

These performances relay on many parameters. Namely, the number of particles used, the number of rays used to simulate the sensors, the validation criterion, the selection method, the redistribution criterion and the initial position of the robot in the environment. In this section, all these parameters will be evaluated using a simulation of the particle filter implemented in Matlab. In all simulations, the robot speed is fixed to $0.4m/s$ and the particle filter iteration is supposed to take $0.1s$. A Matlab code was developed to simulate the particle filter using a specific set of parameters and compare the results for a given trajectory.

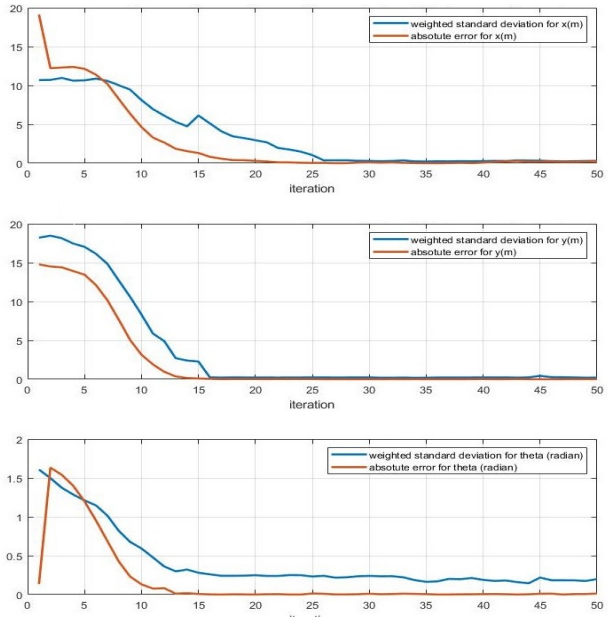
## 3.2 Validation criterion

Since the information of the real robot pose is not available, the particle filter evaluation will be based on the number of iterations that it takes to converge (we suppose that ideally each iteration takes $0.1s$ on the real robot). For that we need first to fix the validation criterion (convergence criterion), i.e, at what iteration the estimated position by the particle filter is supposed to be valid.

For that, multiple approaches can be chosen : The range of particles (the range of $x$, $y$ and $theta$) used in [4], standard deviation and weighted standard deviation that we will study here.




**(a)** Particles range and absolute error




**(b)** Weighted standard deviation and absolute pose error

**Figure 5:** Validation (convergence) criterion study

Figure 5 shows the absolute error with two validation criterion for a given trajectory in the experimental area (CR2 1). First in 5a, the range of the particles in used [4], the idea is that if all particles converge to a pose vector, then, it is probably the actual robot pose. Both range and absolute error converge to 0, but the range took almost 10 iterations to converge after the error convergence, this is due to the existence of some particles far from the robot position (with a small weights).

To improve this, the weighted standard deviation is used. The results in figure -5b shows that both error and weighted standard deviation converge to 0 at almost the same iteration. Thus the selection of this criterion for the validation. The weighted standard deviation is defined as follow:

$$wsd_x = \frac{\sqrt{\sum_{i=1}^{N} w(i)(x(i) - mean_x)^2}}{\frac{(N-1)}{N}\sum_{i=1}^{N} w(i)} \tag{8}$$

Where $wsd_x$ is the weighted standard deviation, $x(i)$ is the position of the particle $i$ and $mean_x$ is the mean of all particles positions. The threshold of convergence, based on figure 5b, can be fixed to $2m$ for position and $0.5rad$ for orientation. i.e., the position estimated by the particle filter is supposed to be valid if

$$\begin{cases} wsd_x < 2m \\ wsd_y < 2m \\ wsd_\theta < 0.5rad \end{cases}$$

Since the absolute error converges to zero when the weighted standard deviation converges to zero, only this criterion will be used to evaluate the particle filter performance in the rest of simulations and we suppose that if the particle filter converges to a pose, that pose is close enough to the real pose of the robot.

## 3.3 Particle Selection method

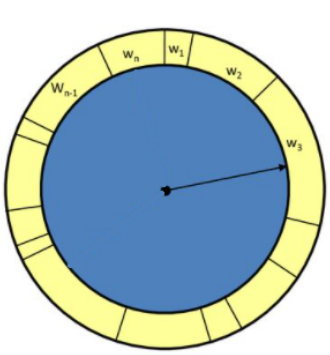
The two methods used for selection are:

### Fitness proportionate selection(roulette wheel)

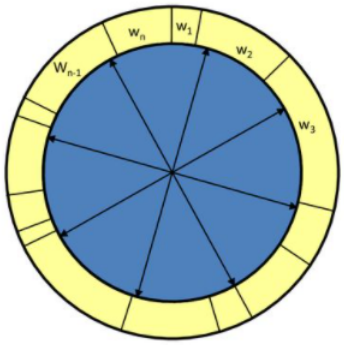
Roulette selection is a stochastic selection method, where the probability for the selection of an individual is proportional to its weight. The method is inspired by real-world roulettes but in this case, the roulette is weighted. The selection is based on the binary search algorithm [8].

### Stochastic universal sampling

This method is a development of fitness proportionate selection (FPS) which exhibits no bias and minimal spread (minimal variance). The previous method choose several solutions from the particles by repeated random sampling. On the other hand, this method uses a single random value to sample all of the solutions by choosing them at evenly spaced intervals. This can give the particles with lower fitness more chance to appear in the next generation.[11] [12]




**(a)** Roulette wheel selection




**(b)** Stochastic universal selection

**Figure 6:** Selection methods

Figure 7 represent a comparison of the particle filter convergence with the two selection methods, using 300 particles, 16 rays, and 14 randomly generated trajectories in the experimental zone (CR2 1), we use the same initial particle distributions since this can affect the results.

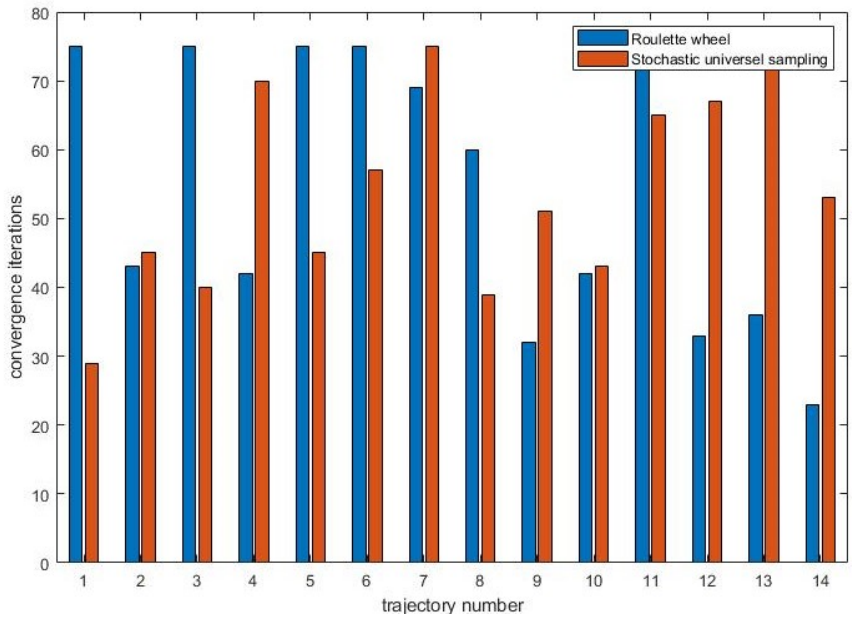



**Figure 7:** Iterations of convergence for both selection method

The results in figure 7 show that the stochastic universal sampling method is generally better than the roulette wheel (the mean number of iteration to convergence is 38 iterations for stochastic universal sampling and 51 iterations for roulette wheel). Adding to that, the roulette wheel completes most simulations without converging due to the selection of

multiple bad particles, increasing the number of particles may solve this since the particle filter will tend to select better particles [7], but since the stochastic universal sampling method is already giving better results, it will be use it in the rest of the simulations.

## 3.4 Number of particles

Increasing the number of particles decrease the number of convergence iterations, but on the other hand, increases the necessary time to execute one iteration.

Figure 8 illustrates the number of iterations to convergence according to the number of particles, using 16 rays and a randomly generated trajectory in the experimental zone.

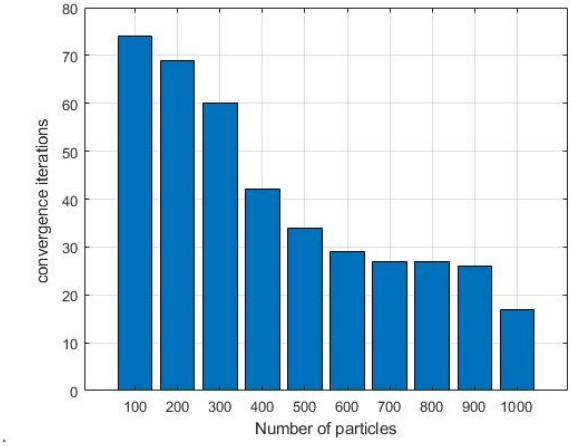



**Figure 8:** The number of iterations To convergence in terms of number of particles

This figure shows that increasing the number of particles does improve the results of convergence, but it highly increases the time of running a single iteration. After studying the number of particles effect on different trajectories, we choose N = 600, this choice generally assures convergence before 40 iterations which is a suitable balance between iterations convergence and iteration running time, (more simulations are present in Annex A2 for that).

## 3.5 Number of rays

While the simulation of laser sensor, the number of used rays affects the performance of the PF, similarly to the number of particles, increasing the

number of rays results in better results and thus the particle filter converge faster but increases the time to run one iteration.

Figure 9 represents the number of iterations to convergence using different number of rays, using 600 particles with the same initial distribution for all simulations, and a randomly generated trajectory in the experimental zone.

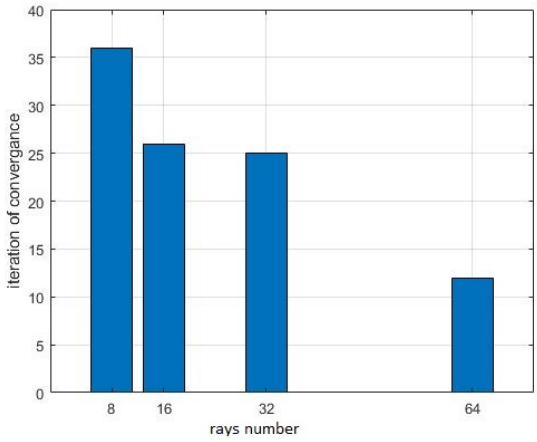



**Figure 9:** The number of iterations to convergence in terms of number of rays

Since increasing the number of rays increase the running time for one iteration and based on figure 9 results, we choose the number of rays to be 16 (more simulations are present in Annex A2 to confirm this choice).

## 3.6 Redistribution criterion

This criterion is very important as explained in 2.2.7 , here the weighted standard deviation is also used since it describes better the distribution of our particles. To fix an appropriate threshold for redistribution, we run the simulation presented in figure 10 using 600 particles, 16 rays and a randomly generated trajectory in the experimental zone.

Based on figure 10, the weighted standard deviation reaches $0.4m$ after 10 iterations after convergence, $0.3m$ after 40 iterations and almost never reaches $0.2m$. Based on these results, the threshold is fixed to $0.3m$. Since, redistributing after only $1s$ is not interesting (each iteration takes $0.1s$) if we take $2 - 4s$ to converge. We do not consider the weighted standard deviation for theta in redistribution since we are only interested in the particle position.

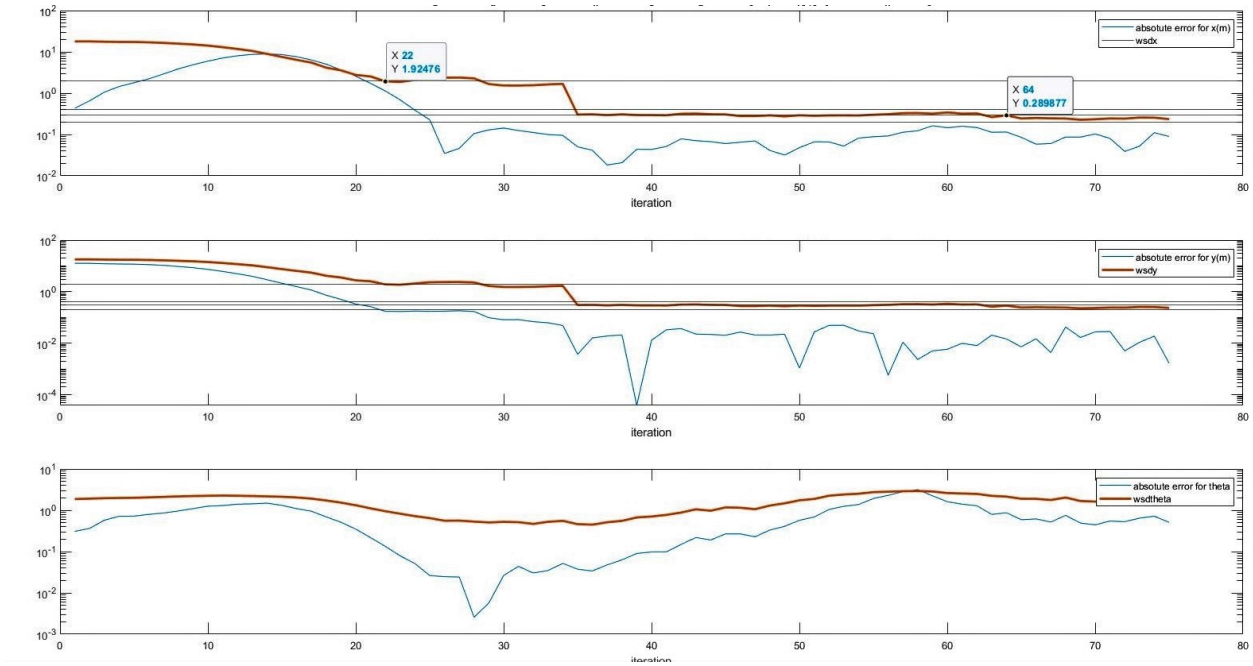



**Figure 10:** The iteration to convergence in terms of number of rays

Another criterion that can be interesting is the maximum weights, since during a kidnapping phenomenon, all the particles will have a bad weights and so the PF will takes a long time to converge and the first proposed criterion won't be enough. That is why, the uses of a weight base criterion can be interesting. We could not do enough simulation to find an appropriate threshold for the weight since it depends also on the position in the environment 1 and the number of particles, thus we didn't use this second criterion but ideally, both should be used.

## 3.7 Conclusion

The goal of this section was to find the optimal combination of particle filter parameters. The weighted standard deviation was chosen for validation and redistribution since it provides all the information on the particle distribution, the stochastic universal sampling was chosen for selection, with 600 particles and 16 rays to simulate the sensors. This set of parameters will be used in the next section to evaluate the performance of the particle filter in all our environment.

# 4 Evaluation and implementation

## 4.1 introduction

The goal of this section, is to evaluate the performance of the particle filter algorithm in both convergence iterations and absolute error in a mission following a given trajectory. After the evaluation, we implemented the particle filter using c++ on the real robot with the goal of running a mission in the experimental zone and comparing the particle filter estimated poses with the real robot poses provided by the QR-codes.

## 4.2 Particle filter Evaluation in the environment

Since particle filter performance depends also on the environment and the start point of the trajectory, in this part, we will take into consideration the set of parameters that we fixed in the previous section to evaluate the particle filter using different trajectories in the environment. The idea is to evaluate the particle filter in different regions of our map. We generate a total of 184 trajectories with a $5m$ distance (most of the trajectories are generated in CR2 since we are more interested in this zone). For this simulation, we fix the number of particles to 200 instead of 600 since 600 particles took too much time to simulate, we use 16 rays and the same first distribution of particles for all the trajectories. The results are presented in figure 11.

Based on this figure, we can see that the PF algorithm generally converges in less than 40 iterations, in some parts of the environment the PF did not perform very well, due to the complexity of the environment in that parts (similarities or the poor performance of the sensors in that area). A similar evaluation is done using the LZA localization method which is another global localization method - a deterministic method based on grid [1], the results (figure 12) show that LZA also performs poorly in the same points on the environment (the same parts where PF didn't perform very well) which confirm that this due to the environmental complexity and not the approach. To solve the problem of lasers in some area, an evaluation for the US sensors is done (results in annex A).

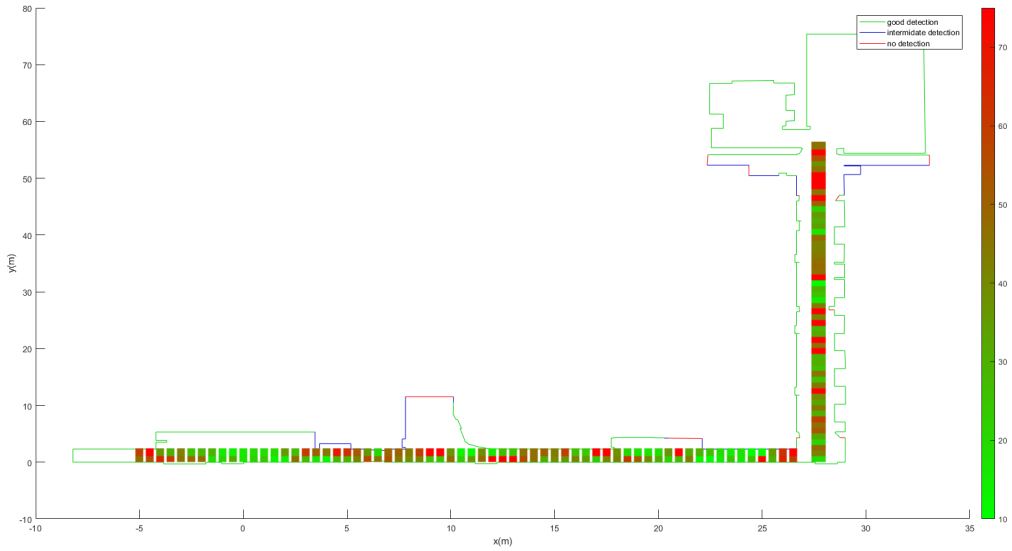



**Figure 11:** Evaluation of particle filter in the environment using laser sensor

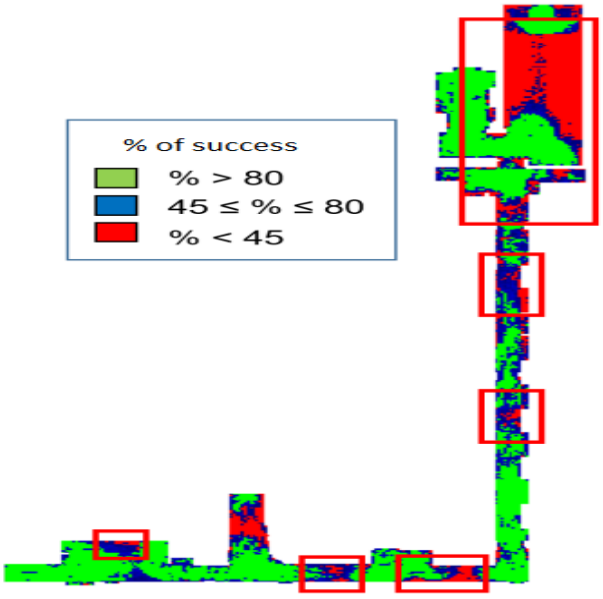



**Figure 12:** Evaluation of LZA in the environment [1]

## 4.3 PF localization performance evaluation

A final mission is done in the experimental zone, in order to evaluate the particle filter performance in term of absolute error on a long trajectory starting from $[x_0, y_0, \theta_0]^T = [11, 1, pi]$ to $[x_f, y_f, \theta_f]^T = [21, 1, pi]$ with a

constant velocity of $0.4m/s$. In this simulation we used 600 particles with 16 rays. figure 13 represent the mission trajectory and The result of the PF are presented in figure 14

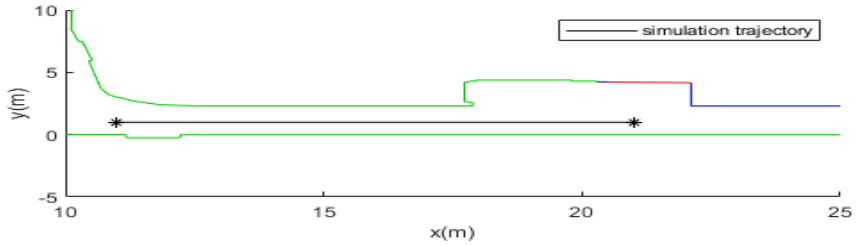



**Figure 13:** mission trajectory

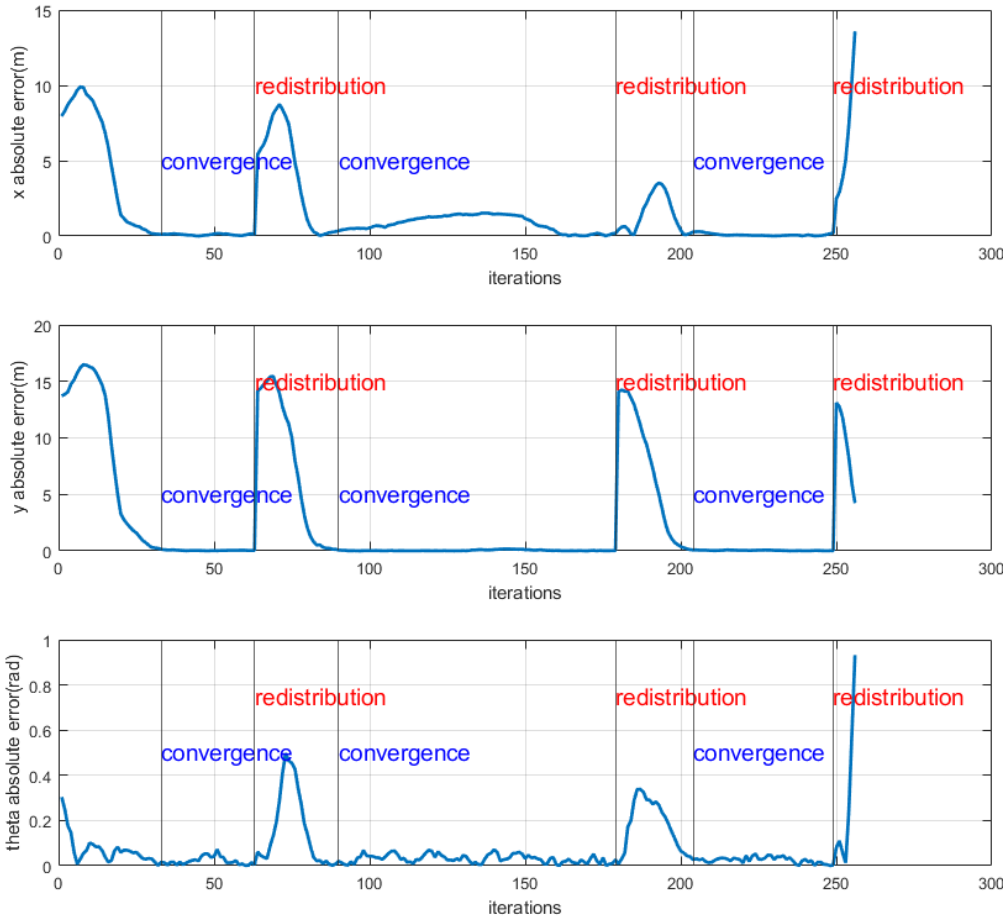



**Figure 14:** Evaluation of particle filter on a long trajectory

Figure 14 shows that the PF generally converge in 20 iterations each time after initialization. On this given trajectory we redistribute the particles three times and each time the particles do find the robot again correctly.

Between $100-150$ iterations $(4-6m)$. The PF perform poorly with $1.5m$ of error this due to the similarities in that part of the corridor (figure 13). But after redistribution, the PF converged again to the correct position which shows again the importance of redistribution.

## 4.4 Implementation

The main mission is to evaluate the Particle filter performances in the experimental area of corridor 2, where QR-Codes are used as landmarks to provide global pose references.

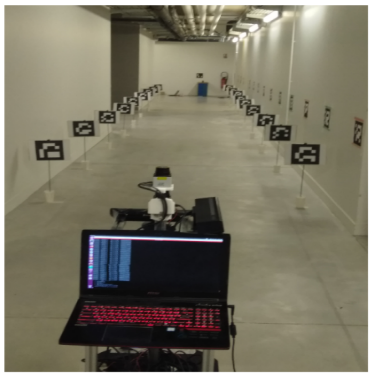


**Figure 15:** The experimental area [1]

The robot mission starts by following a trajectory, using odometry for the control. We run the particle filter in the background without using particle filter estimated poses. After completing the mission, the particle filter estimated poses are validated using QR-Codes location information.

All the implementation code has been developed but we didn't have the time to analyze the results on a mission and compare the results with the QR-Codes location information.

## 4.5 Conclusion

After evaluation and validation of the Particle filter performances on simulation, we aim to implement it on our platform to perform a mission in our experimental area, then compare and validate the obtained results with simulation results.

# 5 General conclusion

Global localization is a deeply investigated field in mobile robotics, and many effective solutions have been proposed. What motivates our work is to evaluate a global localization technique 'Particle filter', in term of convergence performances and adaptability to our environment.

After a first implementation of the particle filter in Matlab, the goal was to enhance the particle filter performances by investing in multiple parameters namely, number of particles, number of rays, selection methods, validation, and redistribution criteria.

Then we evaluated and qualified its performances (in simulation) in terms of uncertainties and adaptability to our environment and compare it with another absolute method LZA.

The implementation of the particle filter in the robot was done, after developing and testing all particle filter functions in c++, but due to time limitations, we couldn't do the mission with the robot.

As perspectives we aim to:

- Use the localization performance guarantee of the particle filter to accomplish the Pioneer-P3Dx mission in our environment, and compare the obtained results with the simulation.

- Further simulations need to be done to evaluate and validate the particle filter performances in the experimental zone.

- Optimize the c++ implementation, which will reduce the time that the robot takes to do one iteration, this will allow increasing the number of particles and rays, which improve the particle filter performance.

# References

[1] Philippe Lambert. *Contribution à l'autonomie des robots: vers des missions à garantie de performance incluant l'incertitude de localisation en environnement intérieur connu*. PhD thesis, Université de Montpellier, 2021.

[2] Stephen Se, David Lowe, and Jim Little. Local and global localization for mobile robots using visual landmarks. IEEE, 2001.

[3] Frank Dellaert and Wolfram Burgard. Carnegie mellon university, institute of computer science iii, university of bonn.

[4] Moussaoui Mahmoud and Belmahi Aboubakr. Localisation par filtrage particulaire. rapport du projet master2. 2021.

[5] Jeong Woo, Young-Joong Kim, Jeong-on Lee, and Myo-Taeg Lim. Localization of mobile robot using particle filter. In *2006 SICE-ICASE International Joint Conference*, pages 3031–3034. IEEE, 2006.

[6] Jongil Lim, SeokJu Lee, Girma Tewolde, and Jaerock Kwon. Indoor localization and navigation for a mobile robot equipped with rotating ultrasonic sensors using a smartphone as the robot's brain. In *2015 IEEE International Conference on Electro/Information Technology (EIT)*, pages 621–625. IEEE, 2015.

[7] Paul Michael Newman. C4b—mobile robotics. *Accessed*, 2:2011.

[8] David filliat. robotique mobile. engineering school. robotique mobile, ensta paristech, 2011, pp.175. ffcel-00655005.

[9] Christian Musso. *Contributions aux méthodes numériques pour le filtrage et la recherche opérationnelle*. PhD thesis, Université de Toulon, 2017.

[10] Yiploon Seow, Renato Miyagusuku, Atsushi Yamashita, and Hajime Asama. Detecting and solving the kidnapped robot problem using laser range finder and wifi signal. In *2017 IEEE international conference on real-time computing and robotics (RCAR)*, pages 303–308. IEEE, 2017.

[11] James E Baker et al. Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the second international conference on genetic algorithms*, volume 206, pages 14–21, 1987.

[12] John J Grefenstette. *Genetic algorithms and their applications: proceedings of the second international conference on genetic algorithms.* Psychology Press, 2013.

# A Annexe

## A.1 Particle filter performance using different sensor types

All the simulations in the report were done using both laser sensors, since we already use this sensor for obstacle avoiding on the robot, in this part we investigate the performance of the particle filter if we use only one laser (to optimize energy consumption), if we use all the US sensors, if we use only the front US sensors (sensor 0 1 2 3 4 5 6 7 figure 3) or if we use the following US sensor (0 2 4 6 8 10 12 14 figure 3) the results are then compared to the results of figure 11.

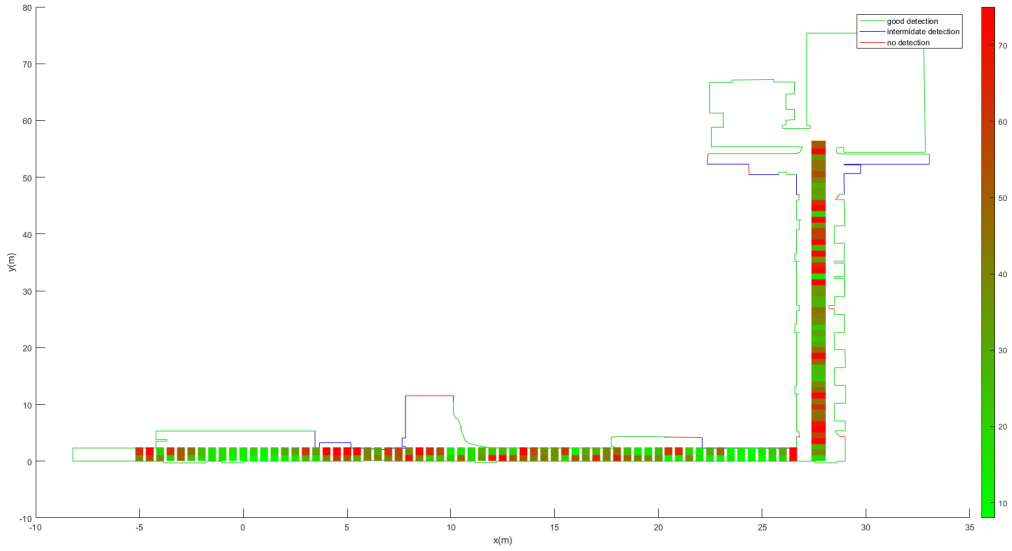



**Figure 16:** Evaluation of particle filter in the environment using only laser 1 sensor

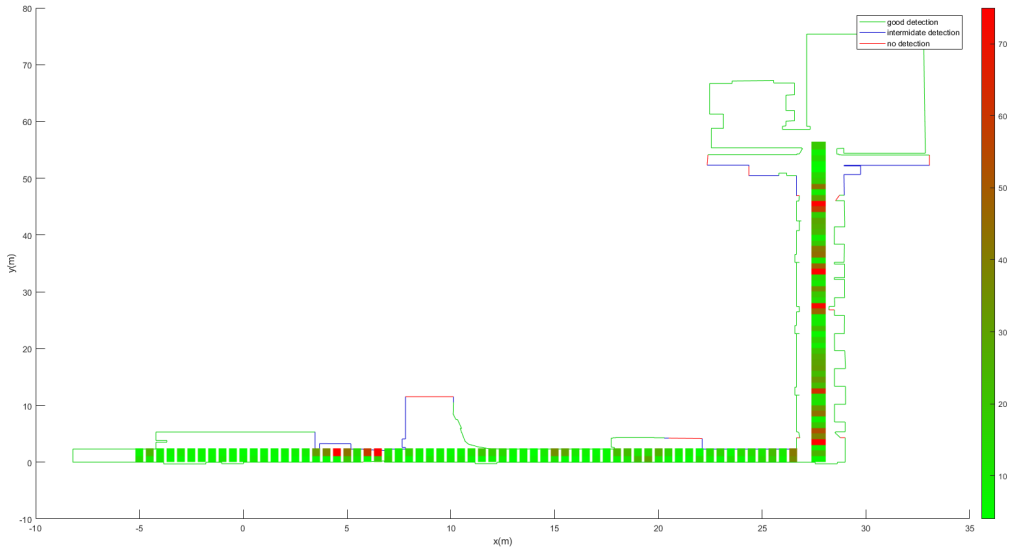



**Figure 17:** Evaluation of particle filter in the environment using US sensors

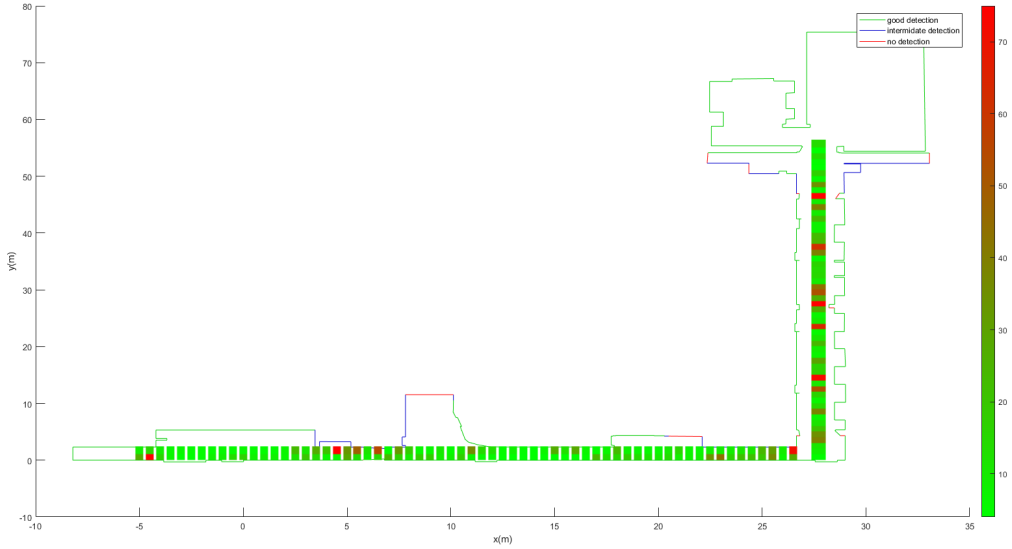



**Figure 18:** Evaluation of particle filter in the environment using the front US sensors

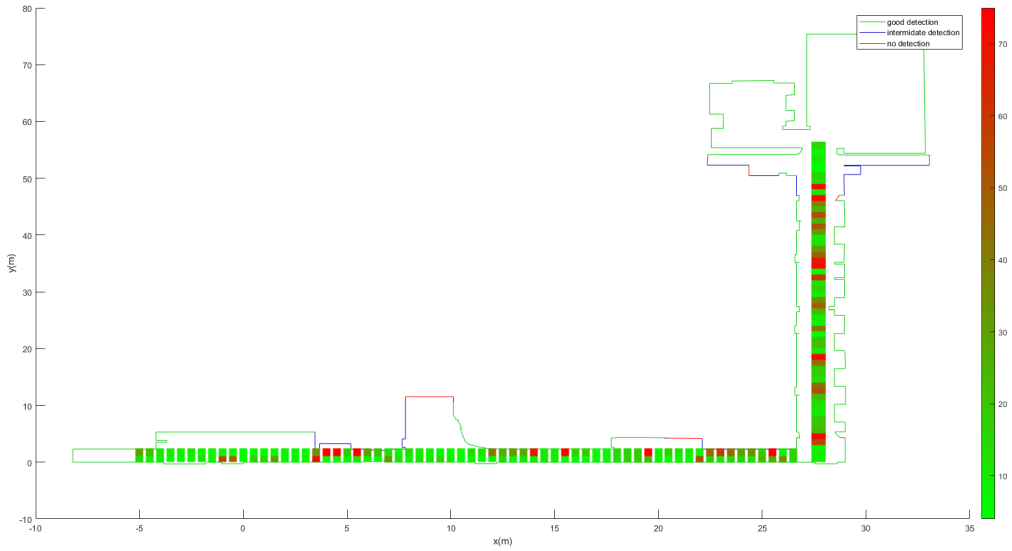



**Figure 19:** Evaluation of particle filter in the environment using laser half of the US sensors

In general, laser 1 results are worse than both laser results, but on some parts of the environment we can converge using only laser 1 in less than 40 iterations. This means that for some parts of the environment we can use only the front laser sensor and turn it off the second sensor to save energy.

US sensors generally give better results and should be used in the parts where the particle filter can't converge using only lasers. In our study, we didn't use US sensors as the first choice for two reasons. First, the energy consumption since we already use lasers on the robot, adding the US sensors will consume more energy that's why we studied the case of not using all US sensors which can be a solution since the results in figure 18 and 19 are still better then the results of the laser sensors, the second reason is that our US sensors don't always give correct readings [1].

## A.2 Particle filter and Number of Particles (additional tests)

The same test of section 3.4 was done using different trajectories to confirm the results and the choice of the number of particles parameter:

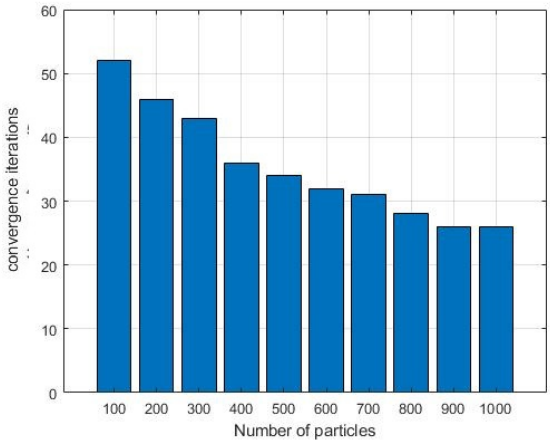



**Figure 20:** The iteration to convergence in terms of number of particles test 2

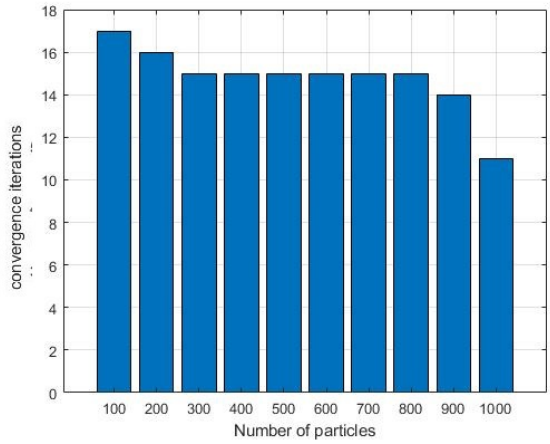



**Figure 21:** The iteration to convergence in terms of number of particles test 3

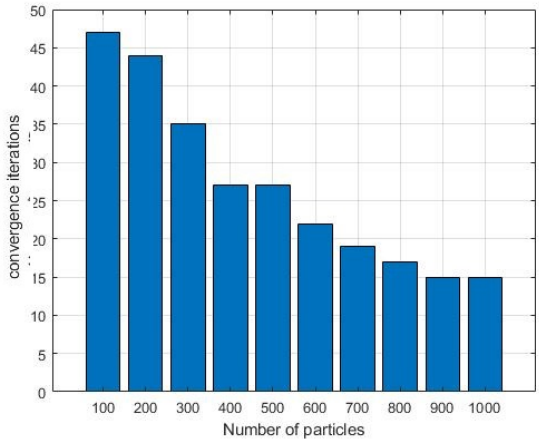



**Figure 22:** The iteration to convergence in terms of number of particles test 4

These results confirm the choice of the number of particles in section 3.4. Indeed, with 600 particles, the particle filter always converges to the desired position in a reasonable number of iterations (less than 40), increasing the number of iterations will increase the time of running one iteration with only a slight improvement in the performance (after reaching a performance of 15-20 convergence iterations figure 21). Also, while using only 100-200 particles, the particle filter is sensitive to the initial distribution of the particles (converge in a range of iteration given by [17-53] in the tests). But, when we increase the number of particles the PF always converge in [12-23]iterations (the same range each time) independently from the initial distribution of the particles.

## A.3 Particle filter and Number of Rays (additional tests)

The same test of section 3.3 was done using different trajectories to confirm the results and the choice of the number of rays parameter:

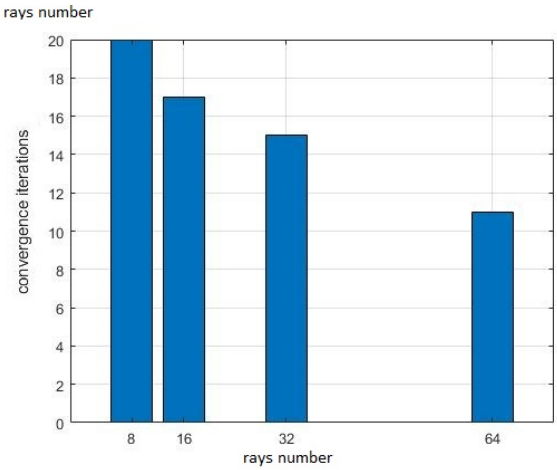



**Figure 23:** The iteration to convergence in terms of number of rays test 2

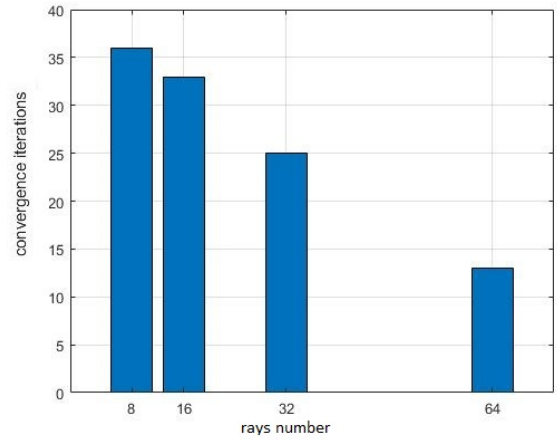



**Figure 24:** The iteration to convergence in terms of number of rays test 3

These results confirm the choice of the number of rays done in section 3.4. With 16 rays, the particle filter always converges in less than 35 iterations, increasing the number of rays will double the time the PF do for one iteration, Thus this choice.