American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS)
ISSN (Print) 2313-4410, ISSN (Online) 2313-4402
© Global Society of Scientific Research and Researchers
http://asrjetsjournal.org/

Localization of Indoor Mobile Robot Using Monte Carlo Localization Algorithm (MCL)

Ali Khaleel Mahmood[a]*, Robert Bicker[b]

[a]Hay al-jamiaa, Baghdad 14001,Iraq
[b]Newcastle University, Address, Newcastle Upon Tyne NE1 7RU, UK
[a]Email: ali_khalil.mahmud@yahoo.com
[b]Email: robert.bicker@newcastle.com

Abstract

One of the challenging issues in robotics is to give a mobile robot the ability to recognize its initial pose (position and orientation) without any human help. In this paper, the components of a mobile robot will be described in addition to the specification of the sensor that will be used. Then, the map of the environment will be defined since it is pre-defined and stored in the memory of the robot. After that, a localization algorithm has been designed, analysed and implemented to develop the ability of a mobile robot to recognize its initial pose. Finally, the final results that have been taken practically will discussed. These result will be divided into two main sub-sections; the first section describes the particles distribution over the working environment and their position update over a number of iterations. Second section will shows the update in the importance weight values over a number of iterations and for three different number of particles.

Keywords: Localization; particles; importance weight; Pose; Differential drive; Kinematic; Global map.

1. Introduction

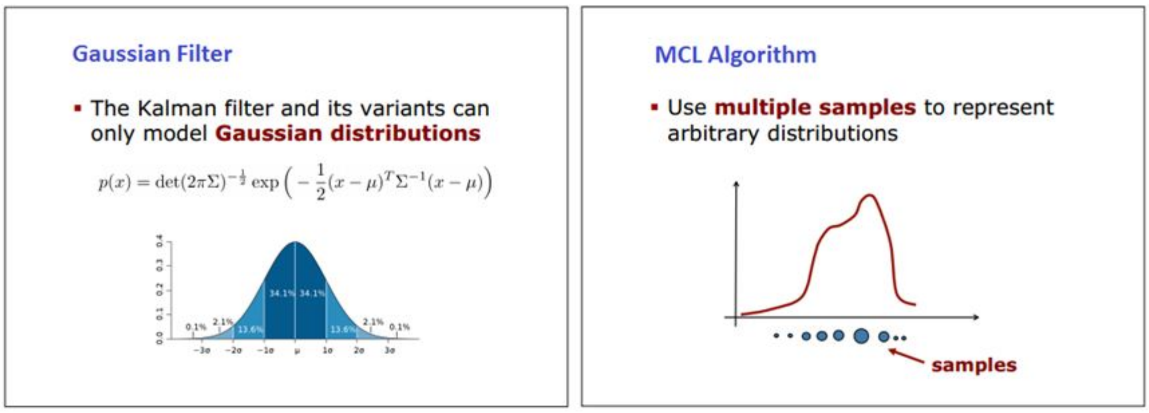
Defining the mobile robot position and orientation within its environment is identified as mobile robot localization problem. Accurate localization of the robot is very important since incorrect estimations for the position and orientation of the mobile robot can cause inaccurate behavior. For example, if a mobile robot does not have an accurate localization algorithm, it will lead to collide with obstacles or objects in the environment or enter the wrong room or even the wrong floor in the building.

----

* Corresponding author.

Thereby preventing the robot from completing its detected goals. For the mobile robot to localize itself within its working environment, it is mostly provided with a map of its environment and it is equipped with sensors to identify itself within its environment. By using the observations (measurements) from the sensors with the map of the environment, the robot determines its position and orientation [1].

In contrast to the position tracking problem, where the starting position and orientation of the mobile robot are known and the robot just needs to correct the small errors in its odometry, the global localization problem will be solved in this project where the robot’s environment is known but the starting point is unknown and the robot has to figure out its position in the environment.


Gaussian Filter

- The Kalman filter and its variants can only model **Gaussian distributions**

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

MCL Algorithm

- Use **multiple samples** to represent arbitrary distributions

Figure 1: Difference between MCL and Kalman Filter

Many localization algorithms have been proposed. Typical examples include Grid localization, Kalman filter and Monte Carlo localization. Grid localization is commonly used for the global localization. The approach used by this method is computing the positional probabilities for each cell in the environment. For this reason, it needs an enormous amount of computation time. Another drawback of this method is that the localization accuracy depends on the size of the cell. In contrast, the Kalman filter technique is generally used in local localization, which uses Gaussian distributions as an error model. According to this technique, the robot continuously estimates its pose by using the sensor data to correct the odometric error. Another popular localization technique used in global localization is Monte Carlo localization (MCL). For global localization, this method is better than grid localization for two reasons: Firstly, because it is less computationally expensive since it compute the probability only for random samples, whose number is much smaller than the cells’ number. Secondly, its result are more accurate than Grid localization since the samples can take any position and direction without any effect by the cell size. In contrast to Kalman filter techniques, Monte Carlo localization have the ability to represent multi-modal distributions so it can globally localize a robot [2].

The aim of this study is to investigate and analyse a localization algorithm for a wheeled mobile robot. In order to satisfy the above aim, pre-defined map should be recognized,analysed and stored in the memory of the robot. Also, a high resolution, high accuracy, and wide angle sensor should be used.

2. Experimental System

In this section, four sections were described. First of all, the physical robot that will be used, NI LabVIEW Robotics Starter Kit. The second section describes the sensor that will be used, Scanning range finder (SOKUIKI sensor) URG-04LX. The third section describes the connection of all the hardware components and finally in the last section, physical environment that will be described in details.

2.1 NI LabVIEW Robotics Starter Kit

The mobile robot used is the NI LabVIEW Robotics Starter Kit. It is a mobile robot platform that features NI Single-Board RIO hardware for embedded control, motors and sensors.

2.1.1 Robot Components

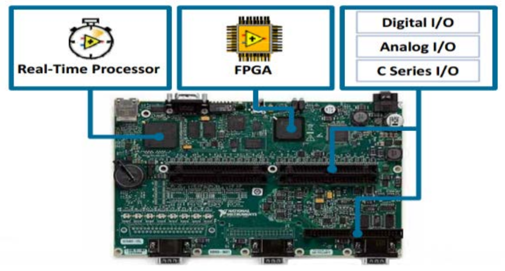


Figure 2: NI single-Board RIO

The NI Robotics Starter Kit uses an NI Single-Board RIO 9631 as an embedded control platform. The Single-Board RIO controller integrates reconfigurable field-programmable gate array (FPGA), real-time processor, and digital and analogue I/O on a single board. The robot has two Pitsco Education TETRIX 4 in (10.16cm) wheels in addition to one Omni- wheel for steering. The robot also has two 12 VDC motors featuring 152 RPM (15.917 rad/s) and torque of 300 Oz-in (2.1185 N.m). For each DC motor on each side of the robot on the front wheels, a 400 PPR (pulse per revolution) encoder is installed. Hence, the motor for each side (left and right motors) can be controlled separately. Also, the robot has PING))) ultrasonic distance sensor. This sensor used to measure distance from 2 cm up to 3 m.

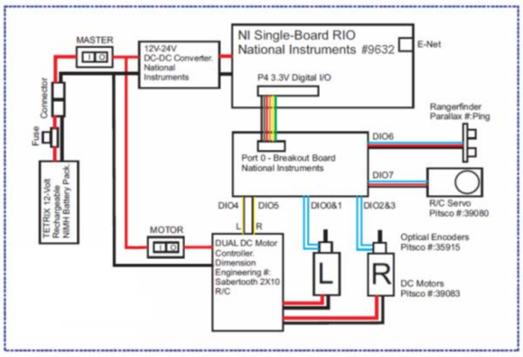

MASTER
12V-24V DC-DC Converter. National Instruments
NI Single-Board RIO National Instruments #9632
E-Net
P4 3.3V Digital I/O
Connector
Fuse
TETRIX 12-Volt Rechargeable NIMH Battery Pack.
Rangerfinder Parallax #:Ping
DIO6
Port 0 - Breakout Board National Instruments
DIO7
R/C Servo Pitsco #:39080
DIO4
DIO5
DIO0&1
DIO2&3
L
R
MOTOR
DUAL DC Motor Controller. Dimension Engineering #: Sabertooth 2X10 R/C
Optical Encoders Pitsco #:35915
DC Motors Pitsco #:39083

Figure 3: NI Robotic Starter Kit 1.0 Hardware Components.

2.1.2 Software Overview

The NI Single-Board RIO 9631 was programmed using NI LabVIEW Robotics Module 2011. National Instrument's LabVIEW Robotics Module 2011 is a graphical programming language that includes blocks for ease of understanding. The blocks involve all the fundamental input and output parameters that are required for different types of functions. LabVIEW in general, has two windows: front panel for representing all the inputs (Controls) and the outputs (Indicators) for providing user control and block diagram for writing the codes. For the Robotic Starter Kit 1.0. Robotics there are built-in drag and drop blocks. These blocks are for initializing the robot, reading / writing for motor control and for ultrasonic sensor control [3].

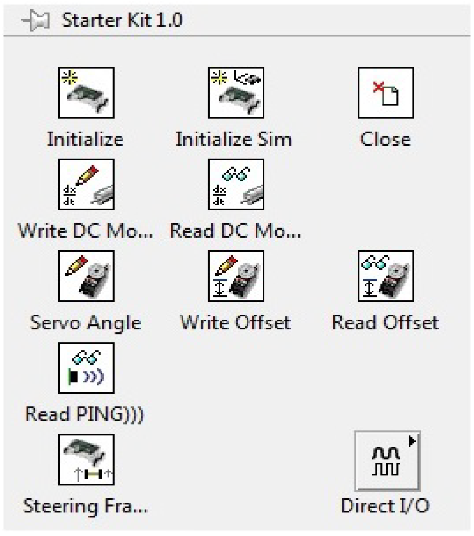


Figure 4: Functional Blocks for NI Robotics Module.

2.2 Scanning range finder (SOKUIKI sensor) URG-04LX

Laser scanners are currently one of the most precise sensors which can be used in sensor-based robot's navigation methods [4]. Also, they have high resolution, high accuracy, and wide angle which supplies the optimum solution for mobile robots.

2.4.1 Sensor Specifications

URG-04LX uses amplitude modulated laser light since the distance to the object can be measuredby measuring the phase shift between the emitted light wave and its reflection. This sensor has been developed by the department of engineering of the Hokuyo Automatic Co., Ltd. and the Intelligent Robot Laboratory of the University of Tsukuba in Japan. As a result of using a brushless spindle motor outer router, it became compact where its size equals to 50x50x70 mm and its total weight 160g.

The distance that can be measured by this sensor is between 2cm and 5.6m with linear resolution of 1mm in addition to 0.36◦ angular resolution because of taking 683 steps on 240◦. In terms of interface and data transfer to the host computer, the URG-04LX has the ability to use two types of connections: Full-speed USB (12Mbps) and serial port RS232 with data transfer rates between 19.2 and 750kbps. The time needed to complete one scan is 100ms because the spindle motor is rotating at 600rpm [5].

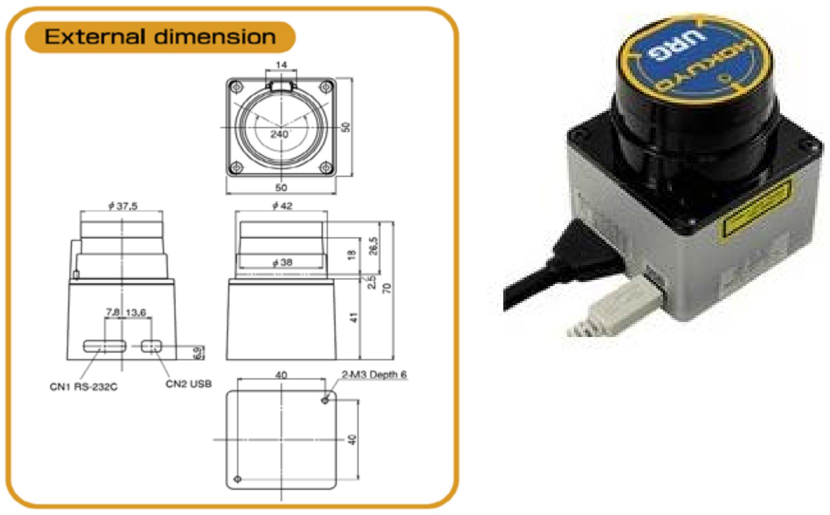


Figure 5: Sensor external dimension

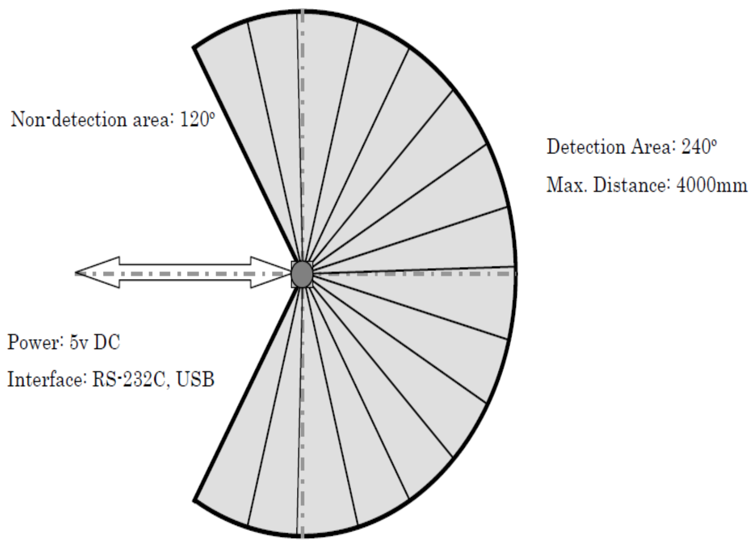


Figure 6: the detectable area of the sensor

2.4.2 Software Overview

Drivers of Hokuyo URG-04LX have been integrated in LabVIEW Robotics. This integration creates an optimum environment for testing and developing many algorithms and applications on laser range finders including autonomous navigation, mapping, path planning and obstacle avoidance.

The SCIP 2.0 communications protocol for Hokuyo's URG family of laser range finders combined with the graphical programming of LabVIEW delivers a fast-track solution for improving and developing autonomous intelligent robotic systems [6].

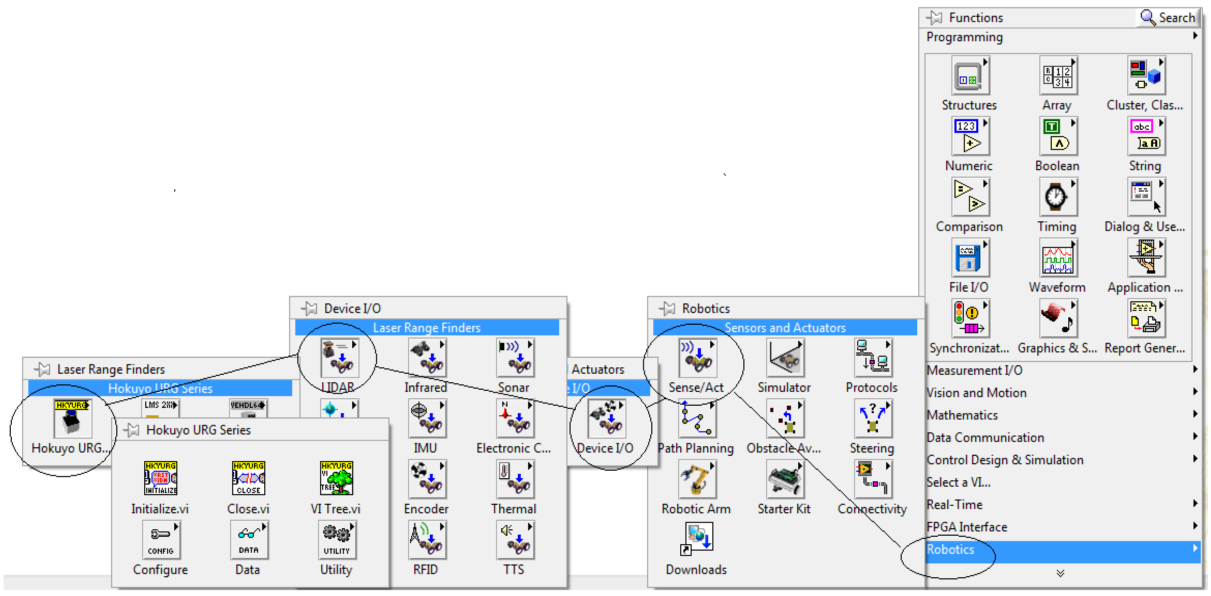


Figure 7: Functional Blocks for Hokuyo URG Series.

In spite of that the sensor has the ability to use full-speed USB (12Mbps), the serial port RS232 with data transfer rates between 19.2 and 750kbps will be used, hence the NI Single-Board RIO 9631 (sbRIO 9631) does not have a USB port. When the serial port is used to communicate with the sbRIO 9631, it is required to provide it with external power (5V DC) hence it does not have the ability to provide the sufficient power to the sensor.

2.3 Connection of the hardware components

It is important to use the optimum way to connect the hardware components together. The first step was connecting the Hokuyo URG-04LX to the NI Starter kit as shown in Figure (8). It is important to consider the alignment of the sensor axes with the robot axes where the difference between them will be considered in the calculations of the distance from the robot to the object.

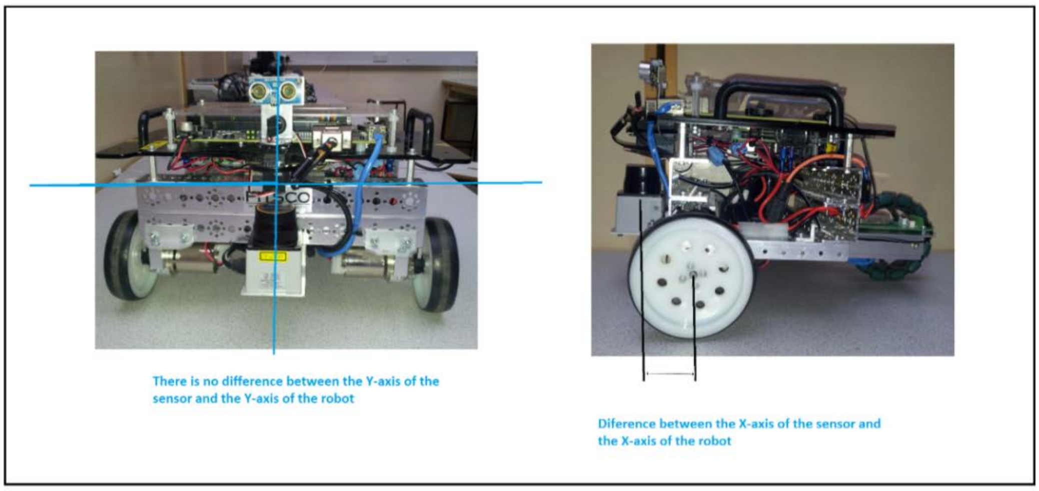

There is no difference between the Y-axis of the sensor and the Y-axis of the robot
Diference between the X-axis of the sensor and the X-axis of the robot

Figure 8: Hokuyo URG-04LX with the NI Starter kit

NI LabVIEW Robotics Starter Kit connected to the laptop using a wireless connection, hence this connection is more sufficient than using wire connection because it will strict the robot movement. The wireless router is used in this project to transmit/ receive date to/from the laptop. For the power requirements, a DC-to-DC convert is used to convert 12V DC battery voltage to 5V DC for the sensor Figure (9).

To supply power with the data from the NI starter kit to the sensor the RS232 modified by adding a power cable to the connector as shown in Figure (10)

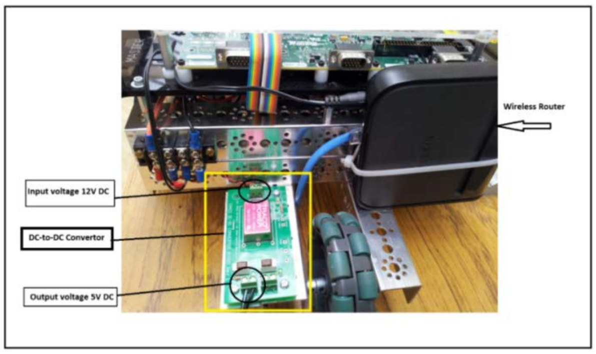


Figure 9: Wireless router and DC-to-DC converter

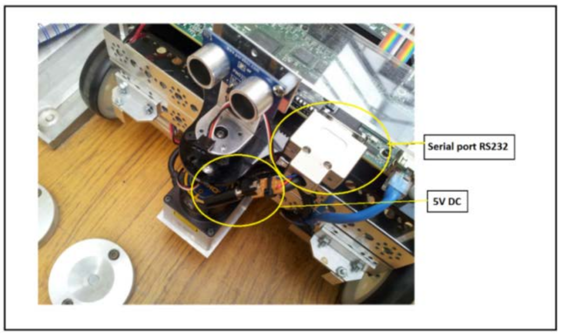


Figure 10: serial port with Power connector

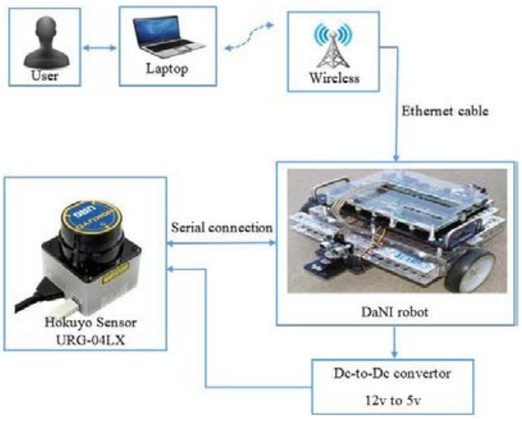


Figure 11: Final Assembly of NI LabVIEW Starter Kit 1.0 with 'Hokuyo URG-04LX sensor'

Finally, the parts will connected together as shown in Figure (11)

2.4 Identification of the Workspace

As in all kinds of sensor navigation it is necessary to define a model map (the global model) to make it possible to define the robot's Localisation. It can be done in several ways. The data concerning the most important obstacles, such as - for example – the walls of rooms, can be introduced from the documentation. The walls on the walls' map that are defined in this way should have the maximum and invariable certainty level. Then, the system creates a bit map of obstacles on this basis. This method gives the best accuracy of the map that is being created, with the assumption that the building was constructed according to plan.

For this project, the map of the environment is pre-defined and stored in the memory of the robot. This section will describe the environment and the method that has been used to represent this environment.

2.4.3 Physical environment

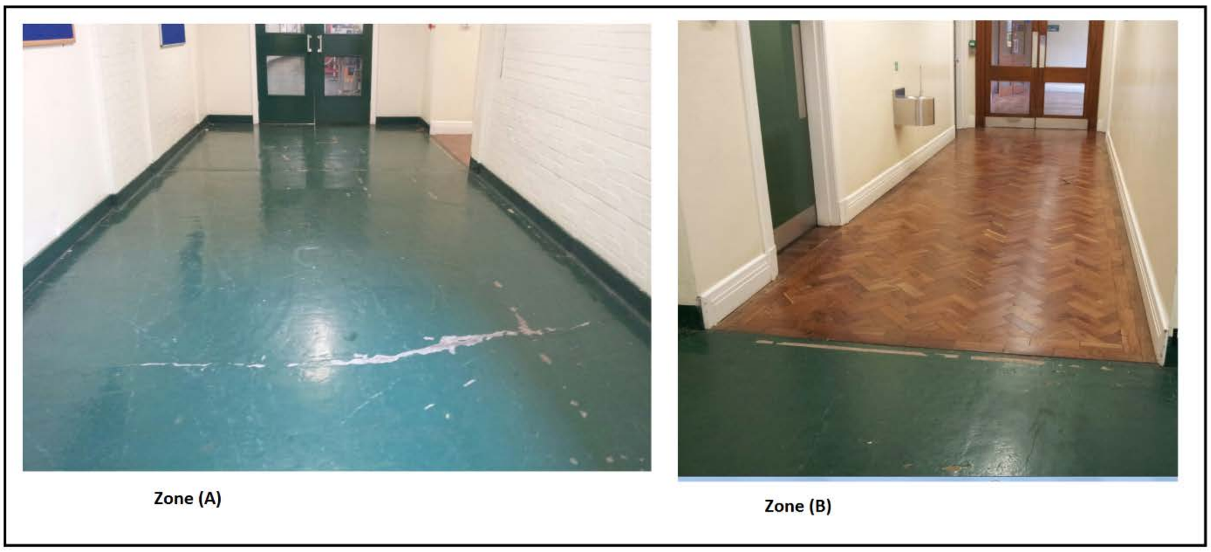


Figure 12: Physical Environment

The environment that will be used in this project can be divided into two zones: Zone (A) and Zone (B) where the size of Zone (A) is 8m× 3.15m while the size of zone (B) is 7m× 2m.

2.4.4 Map representation

There are three fundamental points should be considered when selecting a specific map representation. Firstly, the precision of the map must be chosen appropriately to match the precision of the robot needs to achieve its goals. Secondly, the data types and precision of the map must match the data types and precision returned by the robot’s sensors. Finally, the computational complexity of reasoning about navigation and localization is directly affected by the complexity of the map representation [7].

For this project, grid maps will be used where this type of map representation discretise the environment into grid cells. Each cell has its own information about the area it covers which represented as either occupied or free. The used map is divided into cells with size of 25cm × 25 cm so the total environment size will be (30 cell × 42 cell). The choice of the cell size was depending on both the required accuracy and computational memory where the smaller grid size the higher accuracy but this needs higher computational memory and time consuming. There are two types of data are included in the map which are 1^s and 100^s hence 1^s means that this cell is free and its allowed for the robot to move in it while 100^s means that there is an object in this area and it is not allowed for the robot to move in this area. The aim of using 1^s and 100^s is to be compatible with the path planning algorithm that will discussed in details in chapter 4. The path planning algorithm used to plan a path zbetween the starting point and the target where it will avoid paths that will pass through 100^s.

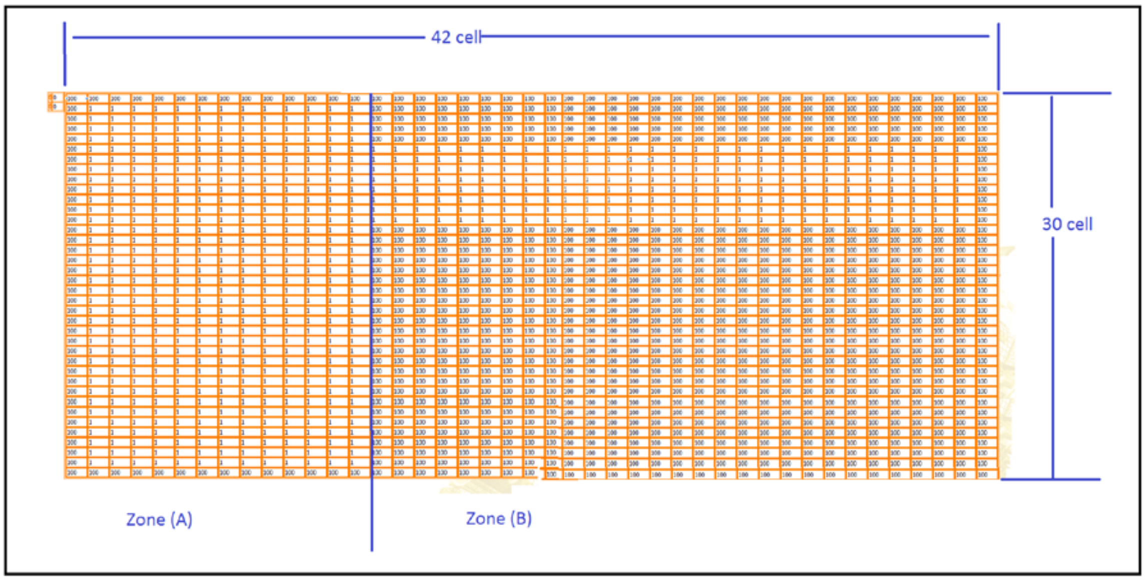


Figure 13: Map representation

3. Monte Carlo localization algorithm (MCL)

Monte Carlo Localization (MCL) is a relatively new method to the problem of mobile robot localization. It estimates a robot's location in a known environment when the sensor and movement readings are given [8]. Among many localization techniques, Monte Carlo localization algorithm has become a valuable and widespread technique in recent years because of having many advantages that are not exist with other localization techniques as discussed previously. MCL using a filter known as particle filter; the role of this filter is representing posteriors by finitely many samples hence it can characterize a much wider space of distributions than Gaussians in addition to modeling nonlinear transformations of random variables .

3.1. Basic Approach of MCL

The MCL technique to localization is depend on a collection of samples (which are normally known as particles). Each particle consist of potential location of the robot currently occupy in addition, these particles have a value that represents the robot probability of currently being at that location. The term 'importance weights' will be used instead of 'probabilities' because, practically, they are not probabilities, they are importance weights'. Although the higher particle number leads the program to converge to a correct solution, it is not recommended to use very high number of particles because they cost more computational time and as a result, the robot movement will be slower.

For this reason that the robot does not know where it is when the program begins, the current particles are uniformly scattered over the range of potential locations and the importance weights for these particles are all the same. Over time, the particles neighbouring the actual current position of the robot should become more anticipated, and those farther away less anticipated. If a line graph was created which plotted the importance weight verses particles, the graph should boost over the actual location. Is it important to notice that it is a flat line in the beginning because the robot does not know its current position.

To build the MCL algorithm, the following steps should be followed:

- Generate a set of particles randomly so that their locations are uniformly scattered also, their importance weights are same.

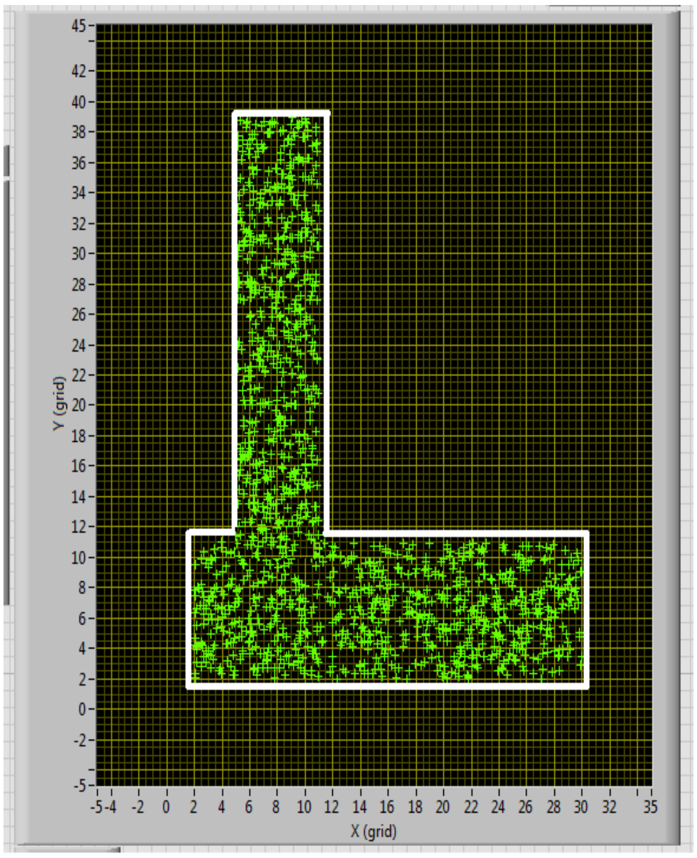


Figure 14: Random Particles generating

- Repeat until done with the current set of samples:

a) Move the robot a fixed distance (motion update), and next take a sensor reading (sensor update).

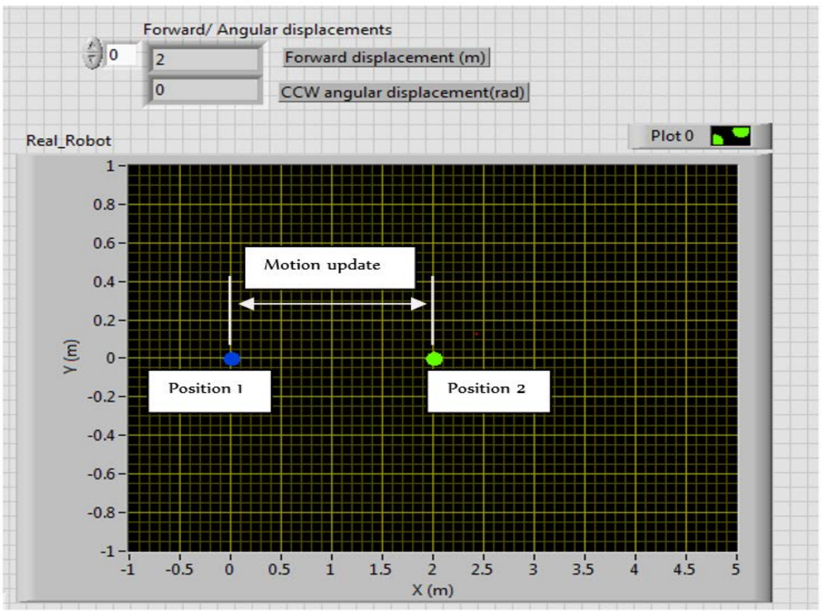


Figure 15: Motion update

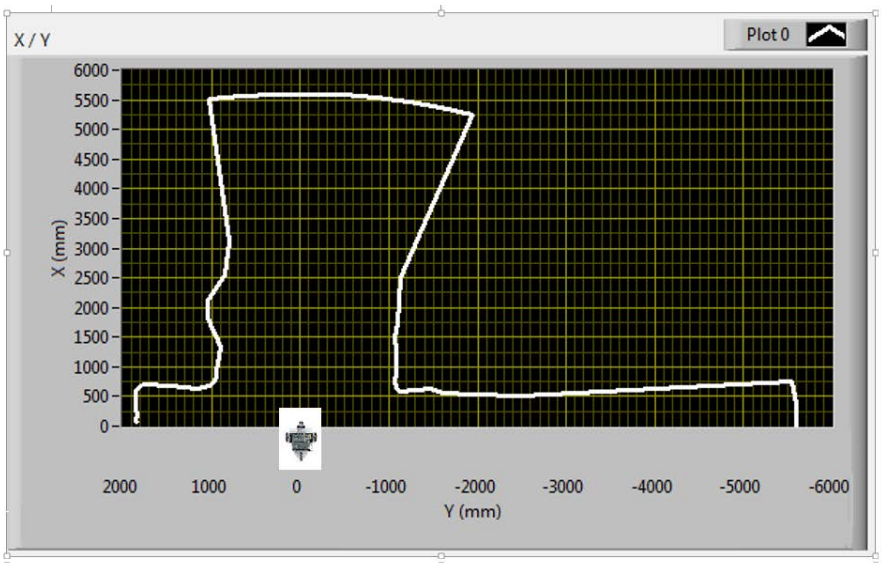


Figure 16: Sensor update

b) pdate the location of each of the particles. This can be done by using the motion update.

c) Appoint the importance weights of each particle to the possibility of that sensor reading (sensor update) given that new location.

It is important to notice that there is a difference between the actual distance and the expected distance so it is essential to add a model for the error to the motion model. Similarly, the sensor seems to indicate a specific object and it might be wrong, and take into consideration; this called sensor model.

d) Create a new collection of particles by sampling and replacing the current set of particles based on their importance weights.

e) Make these new collection of particles become the current set instead of the old particles.

Step (d) consider the most important one. In this step, particles will be chosen depending on their importance weight since the particles with high importance weight are more likely to be chosen than those with low importance weight.

At the end of the program, there will be a set of particles with higher cumulative importance weights than those generated in the start of the program. In addition, it is possible that more than one particle are placed in same location since the estimated location of the robot is in the area with the most particles.

3.2 Basic Monte Carlo Localization Algorithm

For this project, the following parameters will be used [8]:

- N = number of particles.
- St = current set of samples [Xt(i), Wt(i)],
- Xt = current location (X, Y, θ) of the particle (i) at the time t.
- Wt(i) = Importance weight for the particle (i).
- Ut = distance travelled by the mobile robot.
- Zt = current sensor readings

In addition, the number of particle that will used is 1000 particles (N=1000)

```
1) Inputs: Ut, Zt, and particles set St = {[Xt(i), Wt(i)], i= 1, ….., N}
2) For i = 1 to N do                    // this step to update the current set of particles.
a) Xt = update Distance(Xt, Ut)     // This step to compute the new location of the particles
b) Wt(i) = probability( Zt | Xt(i))     // compute the importance weight(probability)
3) St+1 = 0          // Initialize there values then resample to get the next generation of particles
4) For i = 1 to N do
a) Sample an index j from the distribution of particles given by the importance weights in St
b) Add (Xt(j), Xt(j)) to St+1          // Add sample j of the index to the set of new particles St+1
```

5) Return St+1

4. Practical Results

This section includes the final results that have been taken practically. The result will be divided into two main sub-sections; the first section describes the particles distribution over the working environment and their position update over a number of iterations; these results will be repeated with two different number of particles (N=500, 3000). Second section will shows the update in the importance weight values over a number of iterations and for three different number of particles.

4.1 Particles distribution

This section shows the result of two different sets of particles distribution. The first results were taken for N=500 as shown in Figure (17). This number of iterations are relatively small with respect to the environment area so it is expected to get inaccurate results.

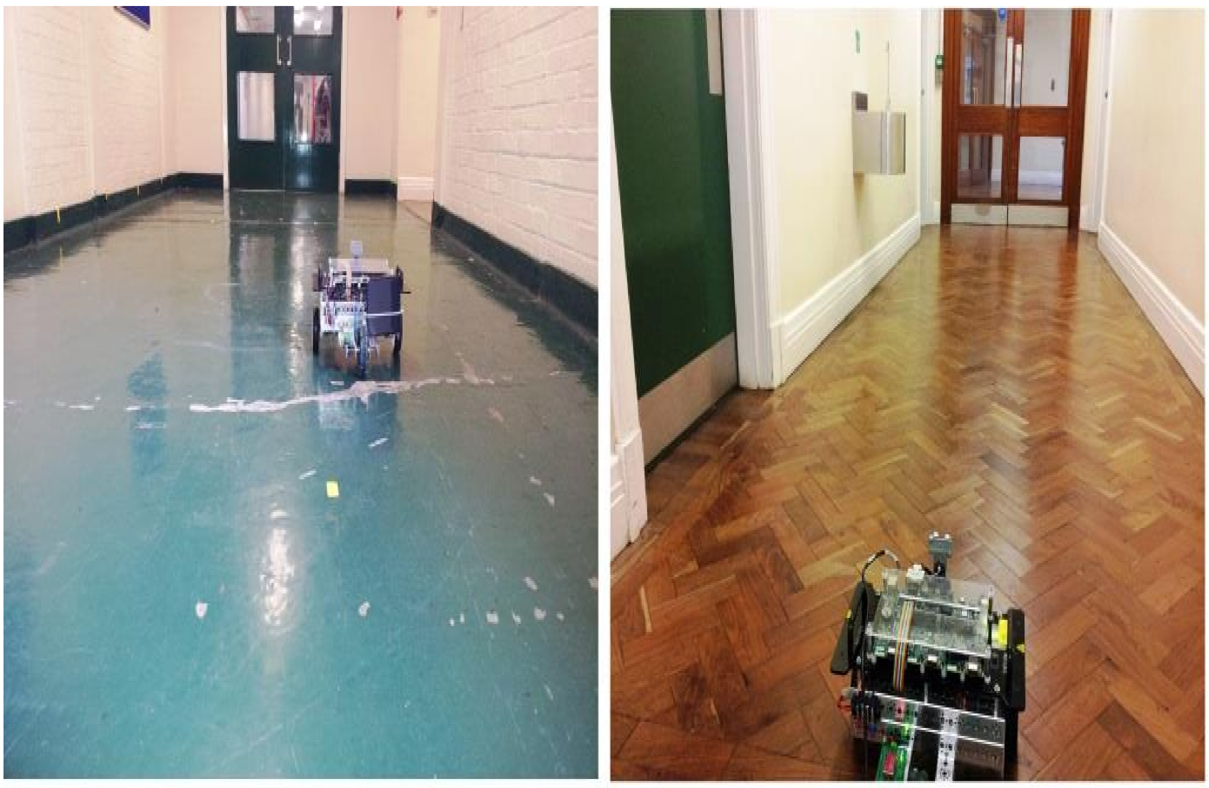


Figure 17: Mobile robot within the Environment

Until the fifth iteration (i =5), the practical distribution is still seems random but, at the seventh iteration, the particles are circulated around the real robot's position. Obviously, the difference between the estimated and real robot positions is still large so, the program will be executed for more iterations. At the tenth and fifteenth iterations, the error became smaller than in the seventh iteration, but is still not acceptable. For this reason, the using of 500 particles is not recommended. For this reason, another test has been done with higher iteration number.

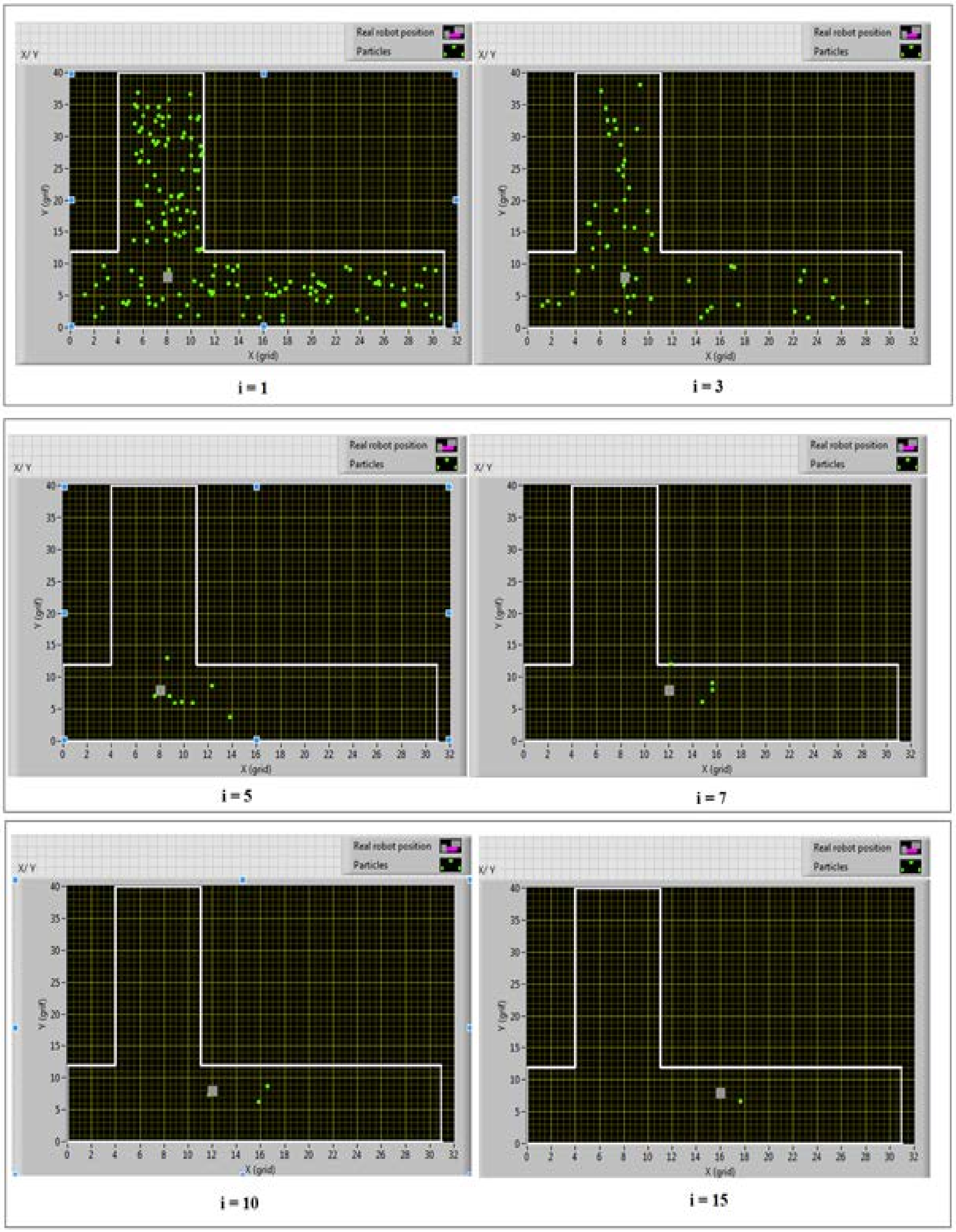


Figure 18: particles distribution for N=500 and i= [1, 3, 5, 7, 10, 15].

The second results were taken for N=3000 as shown in Figure (18). In contrast to the previous results, these results is expected to be more accurate because the number of iterations used in this program is sufficient to give very accurate results.

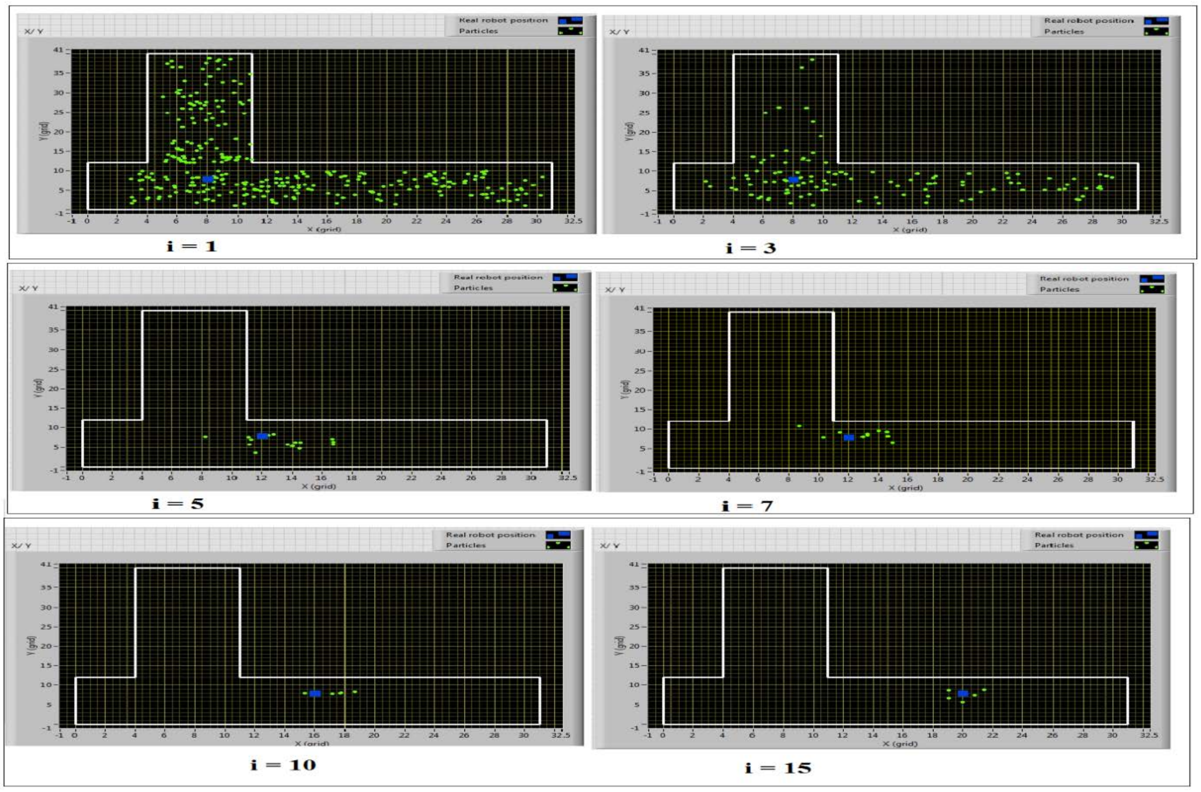
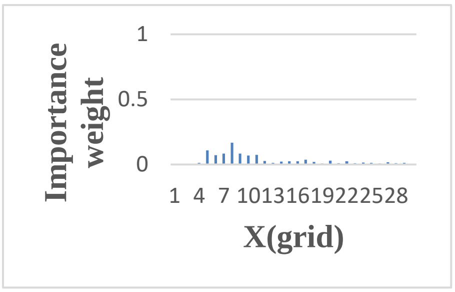


Figure 19: particles distribution for N=3000 and i= [1, 3, 5, 7, 10, 15].

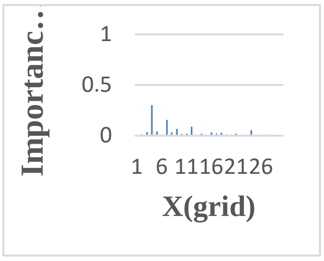
At the first iteration (i=1), the particles started changing their current position but they are still randolmly distributed. Then at third iteration, the practices distribution started to be more seperated, hence the biggest group is around the actual robot position. At the fifth iteration, the particles distribution became around a specific area in the environment which is the real position of the robot. At the seventh iteration, the robot position is almost be detected where the estimated robot position is (Xe =13, Ye = 8) while the real position of the robot is (X =12, Y = 8). This means that the robots needs more iterations to imporve its estimation for the current position; At i = 10, the robot still not sure about its current position. Finally, at the iteration fifteen, the robot detect exactly its current position. Despite that, for this number of iterations, this algorithms takes very long time to be executed (about 20 seconds per iteration), it gives accurate results.

4.2 Importance weights

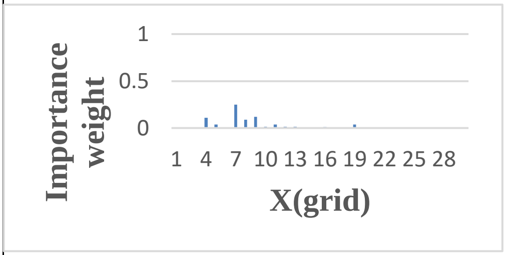
To understand the result that have been presented previously, it is important to discuss the values of the importance weights at each iteration. For this project, the importance weights is measuered with respect to the robot position in the X-axis. This section firstly shows the importance weight for N=500 then ,for N= 3000. It is importance to notice that the summation of the importance weight for each iteration is equal to 1 hence, these valuses are normalized. As a result, thses values of the importance weight can be use as a percentage of the probability of being in that position. For N=500, Figure (19) explains the robot reocgnition for its envorinment hence, at the first iterations the difference between the importance weight valuses is small. In other words, the robot cannot uncertainty about its position is very high. Even after fifteen iterations, the robot wan not able to sellect single value for its postion. So, using 500 itreations can not give correct estimation about the robot position.



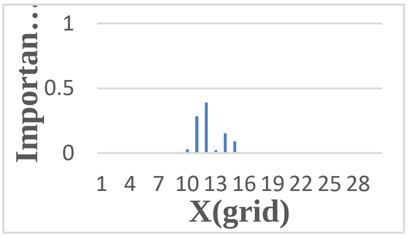
(a) N=500, i=1.



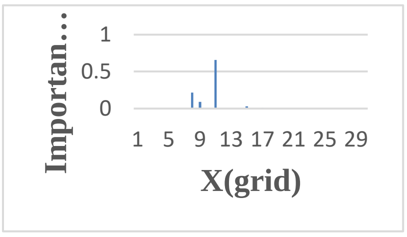
(b) N=500, i=3.



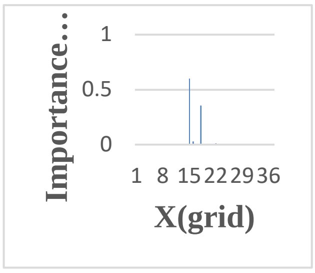
(c) N=500, i=5.



(d) N=500, i=7.



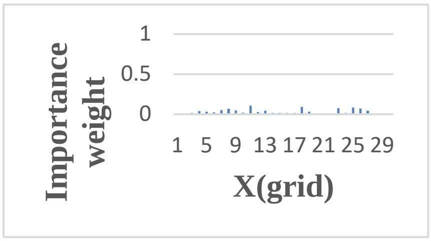
(e) N=500, i=10.



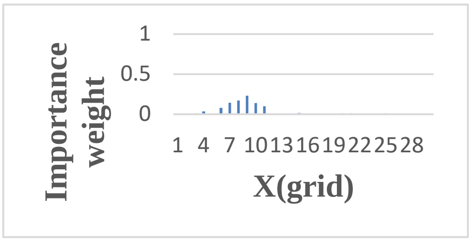
(f) N=500, i=15.

Figure20: Importance weights for N =500

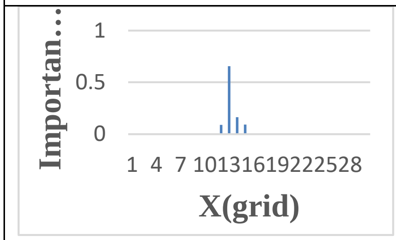
Figure (20) shows the importance weights for N=3000. Obviously, the importance weights values are rapidly increased around the real position of the robot hence, at the tenth iteration, the importance weight at X= 16 is 0.8. This means that the robot is 80% sure about its current position. Finally, at the iteration fifteen, the importance weight is 0.95 at X = 20. In other words, the robot gives estimation of 95% that it is in the position that its X-axis is 20.



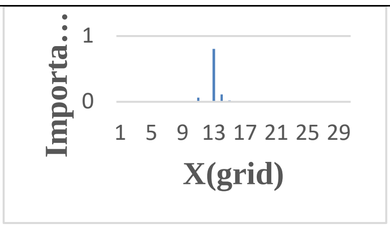
(a) N=3000, i=1.



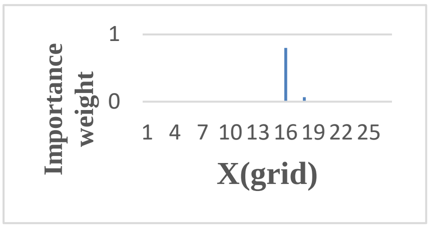
(b) N=3000, i=3.



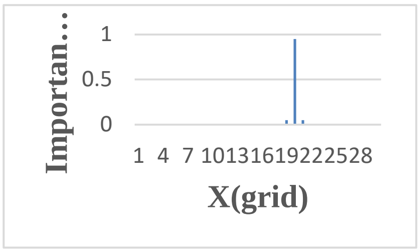
(c) N=3000, i=5.



(d) N=3000, i=7.



(e) N=3000, i=10.



(f) N=3000, i=15.

Figure 21: Importance weights for N =3000

5. Conclusion

This study dealt with the localization of a mobile robot to be able to find its intial position without any human help. This achieved by designing and evaluating an algorithm for localization. A grid maps has been used as a method to represent the map that was pre-defined and stored in the memory of the robot. According to the required accuracy, computional time and computational memory, the the cell size has been selected. Also, Laser scanner sensor has been used which is considered as optimum solution for mobile robots because of its high resolution, high accuracy, and wide angle.

6. Future work

In future work, there are some limitations that deserve further improving:

Long processing time of MCL: Despite that the methods gives an accurate results, the time to execute one iteration is long; the main reason of this is the limitation in the NI starter kit 1.0 (266 MHz only). In the future, it is better to modify MCL to be able to use less particles number but in the same time still giving accurate results.

Failure of MCL: When is no practices around the true location of mobile, or the map is symmetrical, MCL may fail to indicate the correct location of the robot. The problem is that MCL cannot verify whether its result is correct or not. In the future work, more sensor readings will be taken. In addition, adding path tracking algorithm to check if the estimated position of the robot is correct while executing the path planning algorithm.

References

[1] T. Yap Jr " Mobile Robot Navigation with low cost sensors," UNIVERSITY OF CALIFORNIA RIVERSIDE, 2009.

[2] D. Wu, J. Chen,and Y.Wang, "Bring Consciousness to mobile robot being localized," in Robotics and Biomimetics (ROBIO), 2009 IEEE International Conference on, 2009, pp. 741-746

[3] A. Hossian, " Undefined Obstacle Avoidance and Path Planning," 120th ASEE Annual Conference and Exposition 2013.

[4] K. Kozlowski, Robot Motion and control: Springer, 2006.

[5] L.Kneip, F. Tache, G. Caprari, and R. Siegwart, "Characterization of the compact Hokuyo URG-04LX 2D laser range scanner, " in Robotics and Automation, 2009, ICRA'09. IEEE International Conference on 2009, pp. 1447-1454.

[6] E. Ivanjko, I. Komsic, and Petrovic, "Simple off-line odometry Calibration of differential drive mobile robots," in Proceedings of 16th Int. Workshop on Robotics in Alpe-Adria- Damube Region-RAAD, 2007.

[7] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, Introduction to autonomous mobile robots: MIT press, 2011

[8] F. K. Scott Anderson, Pam Lawhead, and Myles McNally, "Monte Carlo Localization" 2004.