Jestr Journal of Engineering Science and Technology Review 9 (3) (2016) 80 – 85

Research Article

JOURNAL OF
Engineering Science and Technology Review

www.jestr.org

Generic Architecture and Localization of a Mobile Robot

Noura Ayadi[1,*], Nicolas Morette[2], Cyril Novales[2], Gérard Poisson[2] and Nabil Derbel[1]

[1]*Electrical Engineering Department, National School of Engineers of Sfax B.P:w.3038, Sfax, Tunisia*
[2]*PRISME, 63, av. Lattre de Tassigny, F-18020 Bourges cedex, France*

Received 24 November 2016; Accepted 24 June 2016

Abstract

In this paper, we aim developing the issue of mobile robot's localization in a known environment. The robot's model and map parameters are initially analyzed and defined. A generic architecture of the system is then presented. The localization algorithm is developed based on the robot's model and data obtained from exteroceptive sensors. A step of optimisation using the Levenberg Marquardt algorithm is then followed. Thereafter, simulation results are described to show the performance of the proposed algorithm. We followed step by step obtained results. Several improvements have been introduced to the algorithm to correct the location process of the mobile robot. Experimental results show that the robot can be tracked with a high accuracy which reflects the efficiency and the reliability of the localization method.

Keywords: Mobile Robot, Generic architecture, Localization, Mapping, Levenberg Marquardt, Simulation.

1. Introduction

This paper presents a combination of different tools in order to create a complete platform to develop robotic applications. We start essentially with the modeling, then the localization method and finally the experimental validation on the wifibot. Described themes are interested in the autonomous navigation of mobile robots, handle cartography, obstacles detection and obstacles avoidance.

Indeed, the autonomous navigation remains the most pressing issue in robotic systems. It depends essentially on the perception, localization, cognition and motion control. All these elements must be well implemented along a control architecture. In the literature, we can find several types of architectures, but the successful one is the most reactive, flexible, robust and efficient. Every architecture should integrate extracted data from robot's sensors, mapping characteristics and actuators capabilities in order to generate intelligent behaviours. The functional architectures are classified into three control categories; the deliberative navigation, the reactive navigation and the hybrid navigation. They represent the combination of the two first types and are known as the cognitive controller [1].

The deliberative control architecture is considered as one of first structures in robotic fields. The user himself must implement the robot's sensors data to fix the desired trajectory. It uses a global model that should be fully or partially known and its static scheme represents navigation limitations in unknown or dynamic environments. The remedy to this problem came with Brooks in 1985, with a reactive navigation control [14]. This is a sub-level structure that helps to deal with complex navigation actions and characterized by a fast response in unknown and dynamic environments. The fact that it doesn't need to know or store the system's model makes it easy to implement and has less computation steps. This ensures robustness and efficiency. Nevertheless, this architecture has a disadvantage with the planning block. Because of the sub-level hierarchy, higher levels are not taken care of while lower levels are being executed. Shortcomings of the two first architectures are overcame thanks to the hybrid navigation control [8]. It integrates a control execution layer placed between a deliberative layer and a reactive layer. Its structure makes it easy to execute the planing block with the deliberative layer and take care of all other blocks in order to avoid failure in case of complex tasks [18].

Once the model is fixed and the system's architecture is defined, the control task is presented. Localization and tracking the robot during its navigation is primordial in order to ensure an autonomous navigation. The purpose of localization is to know the exact position of the robot to navigate without hitting obstacles and detect the goal location to reach [12].

In this paper, the choice of the generic architecture that describes the robot's system among many efficient and functional existing architectures is justified. Then, the localization approach and the mapping procedure aimed to be implemented directly on the Wifibot has been explicited. The proposed localization method is a map-based method. The onboard sensors of the robot is just needed [11, 14]. Finally, we present the simulation results that proves the efficiency and the robustness of the presented algorithm have been presented.

2. Description of the Robot

Wifibot presents a multi-purpose robot. It is a Wifi-enabled, low cost and running Linux platform. Wifibot is useful for

* E-mail address: noura.ayadi85@gmail.com
ISSN: 1791-2377

several applications in the fields of education, research, surveillance and entertainment. It is a differential robot with 4 driving wheels. It is characterized by an open and modular architecture controlled by a serial communication link and by Wifi. It integrates a computing board unit running on Linux, Ubuntu. This platform stands by its simplicity and efficiency.

The Wifibot is composed by an anodized aluminum frame, an USB motorized camera, 4 infrared sensors, central inertia *Vn*−100 and a laser sheet Hokuyo *UTM* −30*LX*. The robot chassis is controlled using a *RS*232 port. The Wifi board ensures a wireless connection of the system with the configured access point which is provided freely. The structure of the Wifibot platform presents an advantage to the user to develop and manipulate different applications in an easy way. The Wifibot robot is presented in Fig 1.

Fig. 1. The Wifibot robot

3. The Proposed Architecture

The application in this work includes several algorithms running in different speeds. Each program is characterized by a specific speed in term of the demanded task. To determine the most suitable systems's architecture, all these programs must be settled in the right way. Capacities constitute term that describes the group of elements that form an architecture. We note as an example: action, adjustment, anticipation, apprenticeship, autonomy, perception, reactivity.

We find in the literature, techniques that ensure best performances of each capacity. However, the problem remains in gathering all these capacities in the functional architecture in a coherent and a wise way. An efficient architecture must be a modular one. It integrates different capacities in particular information about the robot's external environment. It also must integrate deliberative functions allowing to take best decisions.

First architectures are characterized by a downward-way to treat necessary functions in order to develop an action and especially a movement. They are in fact a set of modules joined in series. Interconnection between different modules can not be clearly identified. The structure is presented in Fig 2.

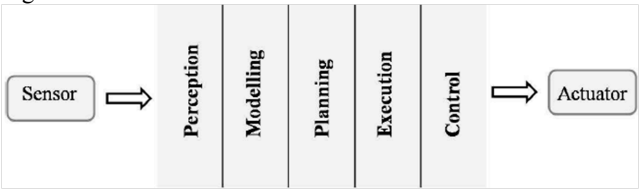

Fig. 2 Traditional structure of control architectures

Since 1986, architectures have evolved into the structure of layers describing necessary functions to control the autonomous navigation of the mobile robot. The most known architectures are subsumption architecture of Brooks [14], the DAMN architecture known as Distributed Architecture for Mobile Navigation of Rosenblatt [10], The ATLANTIS architecture of Gat [5] and the Generic architecture developed in [15]. Layers that form each architecture are interconnected and associated to manage robot's actions. Each one of defined architectures is characterized by a degree of complexity. The defined architecture suitable for robotic applications is the Generic structure which operates effectively in real time comparing the distributed and centralized architectures. In this paper, we propose the use of the generic architecture proposed in [1]. It is composed of three levels. The first one is the fastest characterized by a settling time smaller than 20*ms*. It sends servo information effecting directly motors. The feedback consists only on the information of sensors that pass by a modelling phase before being sent the other levels. The main objective is to define the purpose, the path, the trajectory or system's inputs. The next level is the pilot. It handles the trajectory defined by the navigation phase, it is also capable of managing the reflexed actions and the unforseen events. This loop is fast with a settling time smaller than 1*s* [1, 9]. The higher level is defined by the navigator. It receives the path from the planner and creates the trajectory that the robot should follow. In this level, all kinematic and dynamic restraints are put into consideration. The delivered path from the planner depends on the mission generator which represents a human machine interface, localisation and cartography. Detailed presentation of the chosen structure is described in Fig.3.

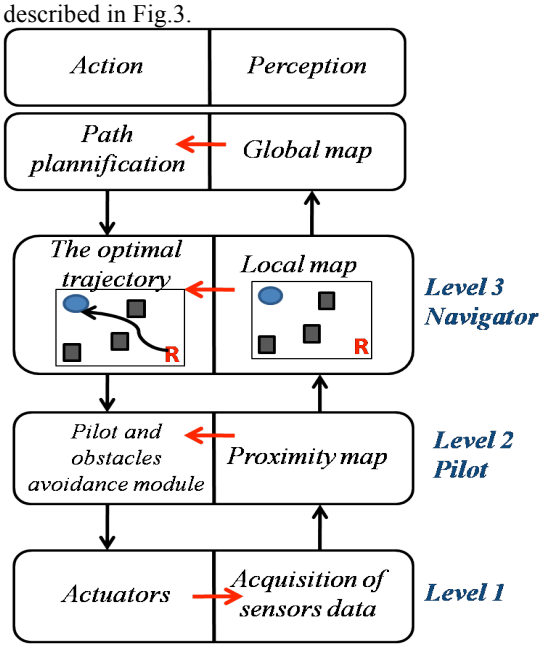

Fig. 3. Generic Arhitecture

The generic architecture has been synthesised at the basis of an omnidirectional robot. In fact, it has proved its effectiveness for other types of robots as classic mobile robots, manipulators and even the remotely operated ones [1, 7, 18].

4. Localization and Robotic Mapping

4.1.Previous work

In recent years, researchers have focused on the localization as it has been considered an important process to know the exact position of the robot during its navigation. They have been based on probabilistic methods that major ones are the Kalman Filter localization, the Markov localization and the particle filtering algorithm also known as the Monte Carlo algorithm.

On the one hand, the Extended Kalman Filter combines results from dead reckoning with extracted data from the exteroceptive sensors [2]. It also assumes that the probability distribution of both the robot configuration and the sensors model is continuous and Gaussian. Subsequently, only the mean value and variance of the Gaussian function are needed to be updated, therefore, the computational cost is very low [17].

On the other hand, the Markov method divides the configuration space into cells. For a mobile robot moving on a plane, the configuration space is 3D dimensional (x, y, θ) and each cell contains the probability of the robot to be in that cell. The probability distribution of the sensors model is also discrete and during action and perception, all the cells are updated [17, 19].

The Monte Carlo Localization (MCL) is derived from the Markov method. Comparing to these previous algorithms, the MCL is easier to implement and ensures higher accuracy. In practice, it shows empirical results by an order of magnitude of memory and computation requirements [4].

4.2.Proposed Method

In this paper, we are addressed to propose an efficient approach in order to locate the robot during its navigation with accuracy. We assume that the robot's initial position is approximatively known. The Inertial Measurement Unit, odometer and laser telemeter are sensors involved in the process. Starting from a fixed orientation, it is due to the inertial unit that we are able to obtain the relative robot's orientation while moving. As to odometric feedback, the robot position can be easily but not precisely calculated. Some errors are caused by interaction of the robot with inevitable features of the environnement as wheel slippage, brutal or fast mouvements.

Realised tests proved that the laser telemeter is the most reliable sensor to obtain an exact position of the robot. It will be considered as the main sensor for the proposed method. The presented method to locate the robot is inspired from the Kalman Filter method where we combine between dead reckoning and extracted data from robot's sensors. Considering the differential structure of the robot and the reliability of the Wifibot's sensors, we are able to obtain the robot's position with a high accuracy. Despite the fact that the robot navigates in a known environment, several constraints may appear especially when the robot must follow a certain trajectory or avoid different obstacles. That is why we need to optimize the localization algorithm. The Levenberg-Marquardt Method (*LMM*) is the most suitable technique to use.

Regarding nonlinear least square problems, the *LMM* tries to fit a parameterized function to a set of measured data points by minimizing the sum of the squares of the errors between the data points and the function. This technique is in fact a combination between the gradient descent method and the Gauss-Newton method considered as standard minimization methods. Using the gradient descent method, the sum of squared errors is reduced by assuming the least squares function which is locally quadratic and finding the minimum of the quadratic. The *LMM* acts like a gradient descent method when parameters are far from their optimum values and acts like the Gauss-Newton method when parameters are close to their optimum values.

However, the algorithm effectiveness depends on local minima which can not be necessarily global ones. This fact, mislead to an incorrect information especially in presence of symmetric effects in the navigation environnement. To avoid this problem, we rely on the approximation of the robot position provided by odometric sensors. Consequently, every time the robot moves, the algorithm initializes its actual position comparing to its previous one. This way and despite the deterministic nature of the *LMM* algorithm, it presents best convergence properties and ensures more precise and quicker solutions.

We define Δ_i the convergence step of the Levenberg-Marquardt algorithm by:

$$\Delta_i = (H_i + \lambda I)^{-1} G_i$$

where H is the Hessian and G_i is the gradient of the cost function for parameters (x, y) and λ is a setting parameter which increases when the cost function diverges [13].

4.3.Application on the Wifibot

We developed a tracking algorithm depending on the robot's position defined by (x_R, y_R) and the orientation θ_R. Initially, we are considering this position while the optimal real one is thereafter given by the *LMM* algorithm. The localization technique is based on the following steps:

- Acquisition of the telemeter measurements
- Computation of the distance between the obstacle and the robot based on its actual position
- Defining the convenient cost function Z for the system
- Minimizing Z using the gradient method

The mapping step requires a known environment where the wifibot navigates in presence of obstacles that it should avoid. Map's dimensions are given as (430×460) cm and an obstacle of (45×30) cm placed in the map at the position (138×286) cm.

4.3.1. Acquisition of the laser rangefinder data

In the localization process, we depend essentially on the laser rangefinder of the Wifibot. Subsequently, we divide the environment into n areas. We note those measurements (m_{t1}, …, m_{tn}).

4.3.2. Distance calculation

Based on the robot's position, we are able to determine n distance values which represent in fact theoretical measures noted (m_1, …, m_n).. We create a loop that increments the variable d until an obstacle occurs. As a result, the obstacle position is defined by this form:

$$x = x_R + d\cos(\theta_R + \theta_C)$$

$$y = y_R + d\sin(\theta_R + \theta_C)$$

where

- (x, y) is the calculated obstacle's position depending on the actual position of the robot (x_R, y_R).
- θ_R is the orientation angle given by the *IMU* of the Wifibot.
- θ_C is a vector containing n values.

The distance d is fixed initially at 0 and incremented in each iteration. In presence of an obstacle, we deduce the measure that we note:

$m = d$

4.3.3. Experimental result

As the Laser sensor supplies 1081 measures sweeping from -135° to $+135^\circ$, we restrained the test to a limited number of samples n. Considering a step of 27°, we obtain $n = 11$ measurements.

Figure 4 illustrates calculated distances in the map in presence of an obstacle and the robot. The robot is situated in the center of the map at the position defined by:

$x_0 = 215cm$

$y_0 = 230cm$

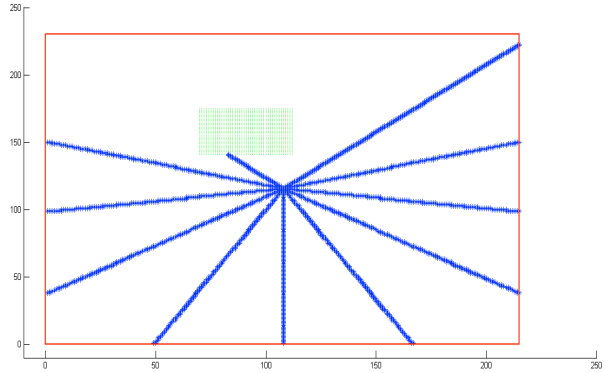

Fig. 4. Calculated distances between the robot and the obstacle

4.3.4 Cost function

We define the cost function as:

$$Z = \sum_{i=1}^{n} (m_{ti} - m_i)^2$$

The main objective is to use the minimal cost function Z in order to locate the robot. We consider the gradient method in this part. To refine the calculation, we integrate in the algorithm a comparison between calculated position coordinates and odometry data that the robot's sensors provide. As a result, coordinates (x_R, y_R) corresponding to the lowest cost function is the most probable real position of the robot.

5. Localization, Simulation Results

In order to obtain simulation results of the localization algorithm, we program the navigation environment of the mobile robot. In *C++*, the map is transformed into an array containing values of {1} to indicate the presence of an obstacle and {0} to show an empty space. The created array is formed of pixels with 1 pixel $=4cm^2$. Figure 5 describes the designed environment.

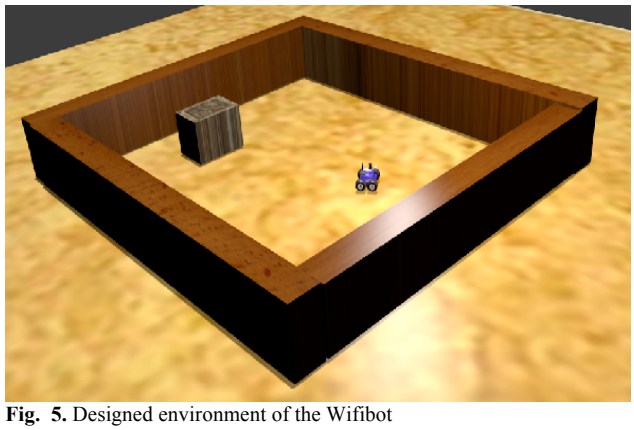

Fig. 5. Designed environment of the Wifibot

5.1. Simulation results

We choose to test the developed algorithm's performance using the simulation calculator MATLAB. The robot is dedicated to reach a target point in the position:

x=50cm
y=50cm

Figure 6 represents the robot's trajectory to reach the target point. In spite of the robot's random initial position, we see that it finds its way to go to its desired destination.

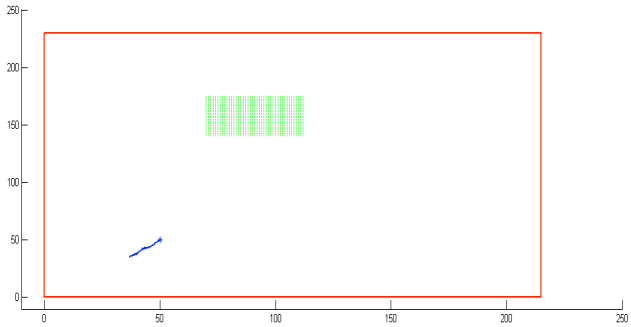

Fig. 6. Trajectory to reach a fixed target

In Fig.7, Fig.8 and Fig.9, we present the position curves (x,y) and the criteria Z that demonstrates clearly that it converges to its minimal value.

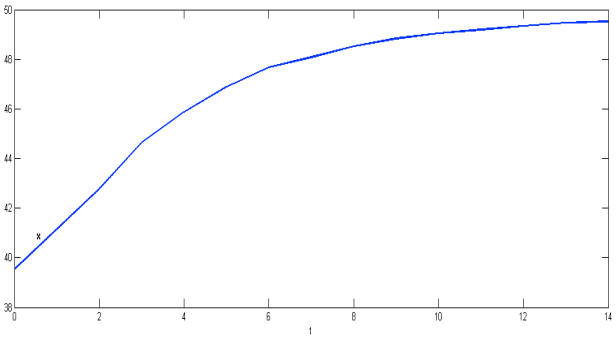

Fig. 7. x trajectory performed through the target

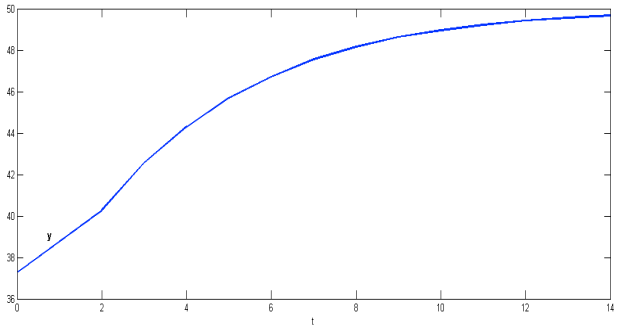

Fig. 8. *y* trajectory performed through the target

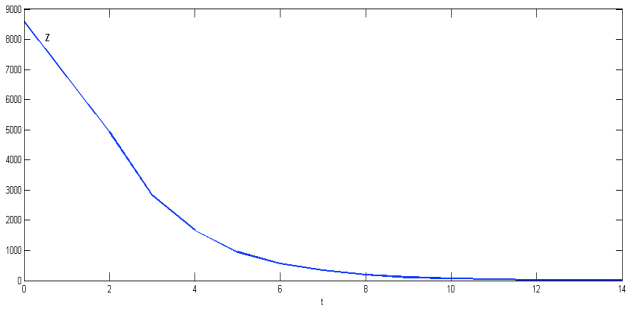

Fig. 9. Convergence of the criteria *Z*

6. Motion Control of the Wifibot

6.1.Introducing the programming tool

After validation of the localization's part of the Wifibot, we look for piloting it and activating the autonomous navigation. The control has been realized using the programming tool RTMaps Studio running on Linux and based on *C++* language. The software is already installed on both Wifibot and the user computer. So, to ensure the communication between them, we developed two projects on each side using appropriate blocks and modules that we call *packages* to send and receive data.

On the robot side, we connect *Wifibot.pck* package which is responsible of sending and receiving information about the system during its navigation. We can access the left and right speeds as well as the odometric position. The laser telemeter and robot's orientation values are also given by two other packages.

On the user side, we created three packages according to the three functions: localization, piloting and navigation. With their *input−output* connections, the modules communicate and permit to localize, control the robot and visualize different trajectories.

6.2.Tests and Improvements

We aim testing the robot's movement. Starting from an initial position, we move it to different points of the map. While observing the laser telemeter's measured values and then the calculated values, we discovered an existing gap that requires corrections and improvements on the algorithm. Every laser beam represents a source of errors when it hits the surfaces of the map. To remedy these anomalies, we proceeded by several methods.

First, we tried to regulate the orientation angle obtained by the *IMU* of the Wifibot with a recalibration of the initial position's angle comparing to nearby angles. The best orientation angle of the robot is the one coinciding with the minimal value of the criteria. However, even if we succeeded to obtain the optimal orientation but errors still exist depending in the way the laser hits the surface. The error is minimal when the laser line is hitting straight the obstacle and more deviation occurs, the error increases. So, we can not trust those values especially that we are not able to detect when measurement errors happen.

Therefore, we created an array called confidence array. It corresponds to the number of real laser telemeter's values. It eliminates every measure exceeding 5 cm comparing to all the nearby measures.

6.3.Simulation Results

We present simulation results of proceeded improvements to the control algorithm of the Wifibot. Fig.10, Fig.11 and Fig.12 are resulted trajectories of the robot in the map in presence of the obstacle. With the blue curve is the real trajectory and the red is the calculated one.

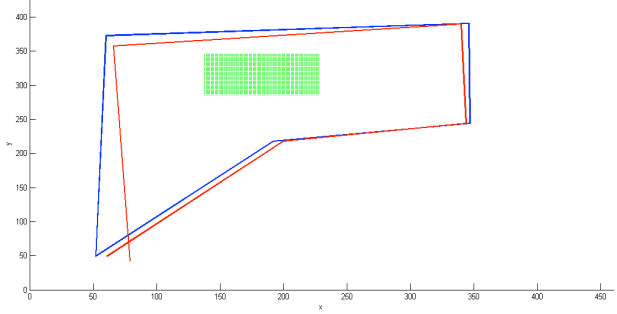

Fig.10. Initial trajectories before corrections

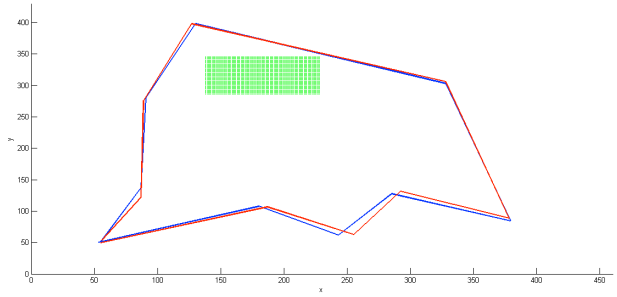

Fig.11. Trajectories after introduction of the confidence array

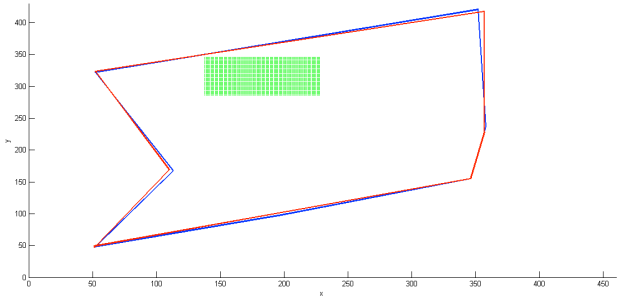

Fig.12. Trajectories after introduction of the confidence array and recalibration of the position coordinates and the angle

The objective of this simulation is to show the ability to track the robot and calculate its position at any time. In addition, it demonstrates the efficiency of the Levenberg Marquardt Method to optimize the robot's trajectory.

7. Conclusions

In this paper, we have proposed the generic architecture to control the Wifibot. This architecture has proved its

efficiency, dynamics and capacity to solve complicated tasks. It has proved to be used for different types of mobile robots. As it is primordial to know the robot's position and orientation all along its navigation, we proposed a method to track the robot and determine the abscissa and ordinate of its control point besides its orientation θ. This method was based on the Wifibot's model and the extracted data from its exteroceptive sensors. We have proceeded then with an optimization using the Levenberg Marquardt algorithm to ensure the convergence of the cost function to its minimal value. Simulation and direct experimental tests on the robot show the robustness of the proposed method and its efficiency to locate the robot in the map. The robot was able to avoid the obstacle and map fronts with success. The next phase is to handle the navigator which tasks are now easier thanks to the localization's robustness.

References

[1] C. Novales, G. Mourioux and G. Poisson, "Une Architecture Modulaire de Commande de Robots: de l'Autonomie à la téléopretaion", Journal Européen des Systèmes Automatisés (JESA), Vol. 5, pp.1-20 (2008).

[2] C. F. Olson, "Probabilistic Self-Localization for Mobile Robots", IEEE Transactions on Robotics and Automation, Vol. 16, No. 1 (2000).

[3] D.W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters", Journal of the Society for Industrial and Applied Mathematics, Vol.11, No.2, pp. 431-441 (1963).

[4] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte Carlo Localization: Efficient Position Estimation forMobile Robots", American Association for Artificial Intelligence, pp. 343-349 (1999).

[5] E. Gat, "Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots", National Conference on Artificial Intelligence (AAAI), pp. 809-815, San Jose, California (1992).

[6] G. Mourioux, Y. Parmantier, C. Novales and G. Poisson, "A wireless omnidirectional robotic plateform for medical use", JJC-LVR, Vol.4, pp. 540-545 (2002).

[7] G. Mourioux, C. Novales and G. Poisson, "A hierarchical architecture to control autonomous robots evolving in an unknown environment", IEEE International Conference on Industrial Technology (ICIT), Vol. 1, pp.72-77 (2004).

[8] H. Yavuz, A. Bradshaw, "A New Conceptual Approach to the Design of Hybrid Control Architecture for Autonomous Mobile Robots" ,Journal of Intelligent and Robotic Systems, Vol.34, No. 1, pp. 1-26 (2002).

[9] J. Canou, G. Mouriowr, C. Novales, G. Poisson, "A local map building process for a reactive navigation of a mobile robot", IEEE international Conference on Robotics and Automation ICRA, New Orleans, Vol.5, pp. 4839-4844 (2004).

[10] J.Rosenblatt, "DAMN: A Distributed Architecture for Mobile Navigation", Spring Symposium on Lessons Learned from Implemented Software Architectures for Physical Agents, AAAI Press, Menlo Park (1995).

[11] K. Nagatani, H. Choset, S. Thrun, "Towards Exactlocalization Without Explicit Localization with the Generalized Voronoi Graph", IEEE International Conference of Robotics and Automation, pp. 342-348, Leuven, Belgium (1998).

[12] M. Pfingsthorn and A. Birk, "Simultaneous Localization and Mapping (SLAM) with Multimodal Probability Distributions", The International Journal of Robotics Research, pp.1-29 (2012).

[13] N.Morette ,C. Novales ,L. Josserand , and P. Vieyres, "Direct Model Navigation issue shifted in the continuous domain by a predictive control approach for mobile robots", International Conference on Robotics and Automation, pp. 2566-2573 (2011).

[14] R. A. Brooks, "A robust layered control system for a mobile robot", IEEE Journal of Robotics and Automation, Vol. 2, No. 1, pp. 14-23 (1986).

[15] R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand, "An architecture for autonomy", The International Journal of Robotics Research, Special Issue on Integrated Architectures for Robot Control and Programming, Vol. 17, No 4, pp. 315-337 (1998).

[16] R. Nadjiasngar and M. Inggs, "Gauss-Newton filtering incorporating Levenberg-Marquardt methods for tracking", Digital Signal Processing, Vol.23, pp. 1662-1667 (2013).

[17] R. Siegwart, D. Scaramuzza, "Autonomous Mobile Robots". Localization and Mapping, Vol. 5, pp.292-338 (2004).

[18] U. A. Sheikh, M. Jamil and Y. Ayaz, "A comparison of various robotic control architectures for autonomous navigation of mobile robots", International Conference on Robotics and Emerging Allied Technologies in Engineering (iCREATE) Islamabad, Pakistan, pp.239-243, April 22-24 (2014).

[19] W. Burgard, D. Fox and S. Thrun, "Active Mobile Robot Localization", International Joint Conference on Artificial Intelligence (IJCAI), pp.1346-1352 (1997).