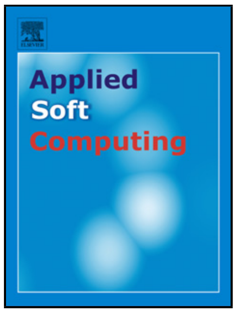


Title: Fuzzy Logic Controllers Design For Omnidirectionnal Mobile Robot Navigation

Author: Mohamed Slim Masmoudi Najla Krichen Mohamed Masmoudi Nabil Derbel

PII: S1568-4946(16)30459-8
DOI: http://dx.doi.org/doi:10.1016/j.asoc.2016.08.057
Reference: ASOC 3799

To appear in: *Applied Soft Computing*

Received date: 21-2-2016
Revised date: 8-7-2016
Accepted date: 29-8-2016

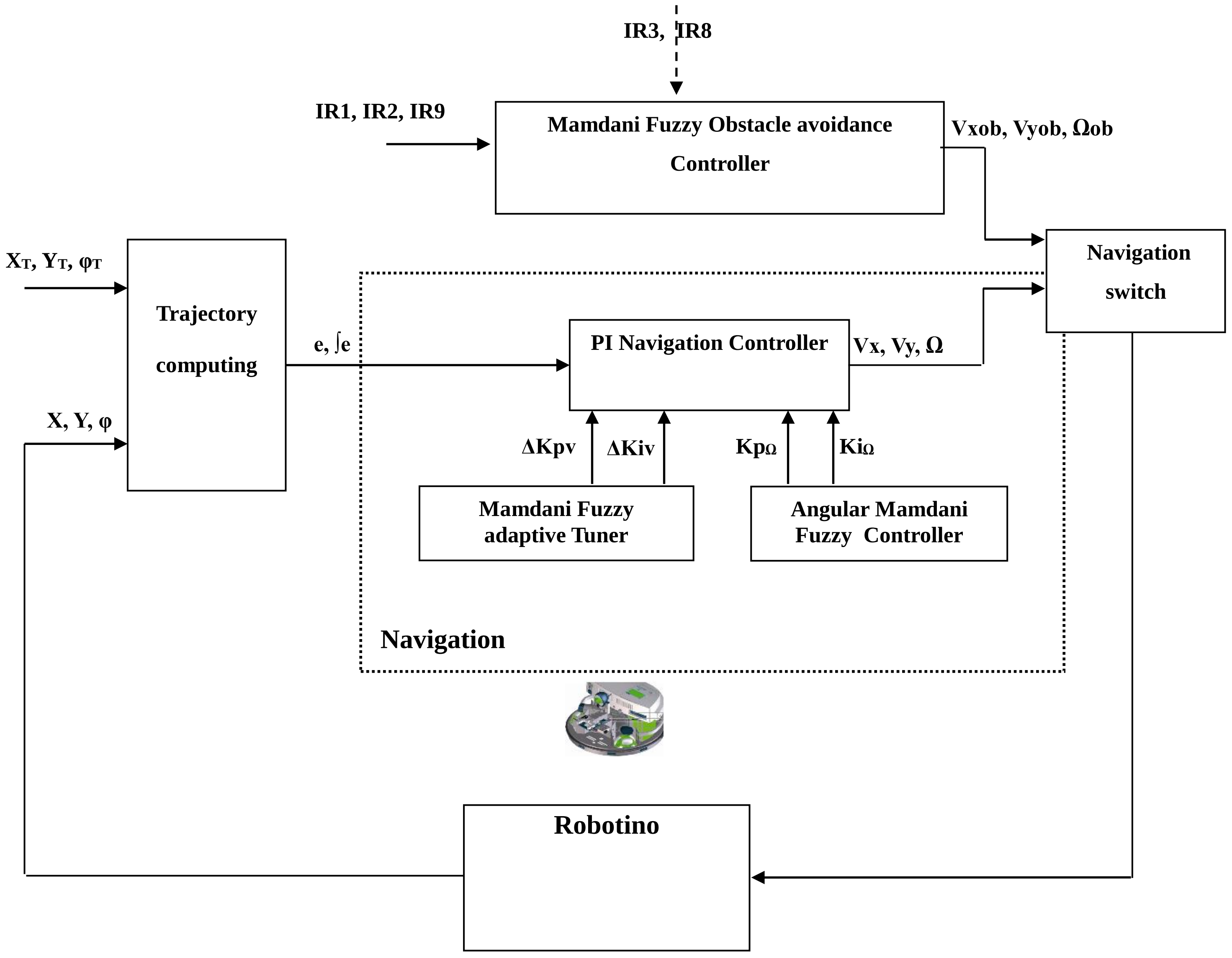
Fuzzy Logic Controllers Design For Omnidirectionnal Mobile Robot Navigation

Mohamed Slim Masmoudi[1], Najla Krichen[2], Mohamed Masmoudi[3], Nabil Derbel[4]

[1]METS Research Group-National School of Electronic and Communications,
Sfax University, Tunisia; email: {med_slim_mas@yahoo.fr}
[2]METS Research Group- National Engineers School of Sfax,
Sfax University, Tunisia; email: { najla_krichen @yahoo.fr}
[3]METS Research Group- National Engineers School of Sfax,
Sfax University, Tunisia; email: { mohamed.masmoudi@enis.rnu.tn }
[4]CEM Lab-National Engineers School of Sfax,
Sfax University, Tunisia; email: { n.derbel@enis.rnu.tn}

Graphical abstracts

In this part, we present the three main contribution of the paper:
- The Fuzzy-PI for linear velocity
- The Fuzzy-PI for angular speed
- Obstace avoidance controller based on Mamdani-fuzzy controller which allows the omnidirectionnal mobile robot to avoid fixed obstacles in an unknown environment.



Highlight

The main goal of the work is to navigate Robotino to reach the target in response to different objectives:
- The robot must reach its target with a desired final angle
-Minimum traveled distance
- An obstacle avoidance algorithm.
To achieve these objectives, a kinematic model details of the Robotino system is given and a navigation Fuzzy-PI controller has been developed. Simulation results, using MATLAB and Robotino-Sim platform, have shown that Robotino is capable to reach the target based on the Fuzzy-PI controller, with a desired goal position and orientation angle. However two limitations have appeared: an error on the final robot angle and much longer travelled distance. To overcome these limitations, we have proposed a solution based on the replacement of the previous intelligent navigation system by two independent controllers. Indeed, we have maintained the previous controller to control the robot linear velocities and we have designed a second fuzzy system to control the robot angular velocity. Adaptive Fuzzy-PI controllers can adjust their parameters to decrease the error caused by the dynamics changes and navigation challenges of Robotino. To achieve the navigation task, we have developed an obstacle avoidance system based on fuzzy logic. Hence, using the proposed navigation system based on three fuzzy controllers, simulation and experimental results have shown a clear improvement of the navigation system performances. Thus, the fuzzy implemented controllers have shown its adaptation to the dynamics of real omnidirectional robots. In fact, Robotino is capable to reach the target position with a desired final angle direction and to avoid obstacles based on the intelligent controllers. Moreover, the proposed intelligent controllers can be applied to other nonlinear dynamics process such as ITS or omnidirectional wheelchair system

Mohamed Slim Masmoudi

Abstract

This paper presents a new design approach for an intelligent navigation algorithm for omnidirectional mobile robots. Unlike the previous works dealing with the navigation of omnidirectional robots that are focused on only the estimation of the final position, the main contribution of the present study is summed up in the fact that the robot has to reach the final desired position with a predefined final steering angle. This latter improvement is a part of researches carried out on Intelligent Transport Systems (ITS). Taking into consideration the drawbacks of proportional integral (PI) control when applied to omnidirectional robot navigation, we develop an approach to design a fuzzy logic PI controller (Fuzzy-PI).
Preliminary simulation and experimental results using the Fuzzy-PI controller have shown limitations as the robot performed a larger path when the desired final angle increased. Thus, a deepen study have concluded that the Fuzzy-PI system cannot control at the same time the linear and angular velocities. To overcome these drawbacks, we propose to replace the previous intelligent navigation system by two independent controllers. The designed Fuzzy-PI controllers can adjust their parameters (K_P, K_I) to reduce the error caused by the dynamic changes and navigation challenges of omnidirectional robot. A navigation algorithm cannot be efficient without obstacle avoidance system. To achieve this goal, we have developed a third fuzzy controller to fulfill this task.
To evaluate the real performances of fuzzy controllers for navigation and obstacles avoidance, simulation and experimental tests are performed for one, two and three obstacles. The obtained results confirm that the omnidirectional robot could navigate in an unstructured and unknown obstacles environment with better performances and efficiency.

Key Words

Omnidirectional robot, kinematic model, navigation, Fuzzy-PI, optimization, obstacle avoidance.

1. Introduction

Fuzzy logic theory is a powerful soft computing technique to control complex and non-linear systems based on human expert knowledge. In this paper, the navigation and obstacles avoidance problems of a Robotino omnidirectionnel robot is addressed using different fuzzy-logic control algorithms. Currently, omnidirectional vehicles are becoming increasingly popular in both academic and industrial fields, e.g. mobile robots. The latter have some distinguishing advantages over nonholonomic mobile robots. With its three degrees of freedom, omnidirectional mobile mechanism has the great ability to move at each instant in any direction without reorientation. The navigation control of the mobile robot can be classified in two main types: trajectory or path planning's. The first one generates a path between two points with eventually a collision avoiding system. The second type tries to follow the path generated from à predefined lane with a minimal error. However, the navigation of omnidirectionnel robots is a very challenging problem. In fact, classical approaches do not work appropriately since they need the analytic model of the system. However, this modeling is not always available and sometimes impossible to formulate. One of the classical controllers is the famous PID algorithms used for robot control. However, the major drawback of this conventional controller is the need of its parameters estimation based on system model. These parameters rely on the system identification conditions such as the operating point, or if the plant parameters changed. In this case, PID parameters cannot compensate the alterations adaptively [1]. In order to overcome these problems, soft-computing systems are considered as a powerful tool to control a mobile robot [1]-[7]. Fuzzy-logic is considered as a good technique to solve problems, dealing with imprecise aspect and when human expert based-knowledge is provided, making it an adequate theory to develop an omnidirectionnal robot navigation algorithm. Previous academic works have used the fuzzy-logic theory to design a Robotino navigation control algorithm [8]-[9]. Robot navigation performances can be improved by developing a fuzzy adaptive PI-controller that combines the advances of conventional PI with the fuzzy logic theory [10]-[12].

Based on these previous literature works, one could state that PID controller combined with fuzzy-logic is considered to be an effective soft computing technique to control omnidirectionnal robot navigation. Fuzzy-PI controller is a powerful algorithm that can be used when the system is difficult to model, and since the conventional control technique does not provide the expected results even when expert knowledge's are available.

The main requirements for Robotino navigation, addressed in this work, are the following: (i) reach the target position with a desired final angle, (ii) minimize the traveled distance and (iii) avoid occurred obstacles. To fulfill these requirements, a Fuzzy-PI controller was firstly designed and tested. However, the obtained navigation performances confirmed that the same fuzzy-controller could not be used for controlling together the linear and angular velocities. To overcome these drawbacks, we propose to replace the previous intelligent navigation system by two independent Fuzzy-PI controllers. Moreover, to achieve the navigation requirements, we propose to develop an obstacle avoidance system using a third fuzzy logic controller.

The paper is organized as follows. Section 2 introduces the kinematic modeling of the omnidirectional robot. Section 3 details Fuzzy adaptive PI controller architecture and simulation results obtained using Matlab and Robotino-Sim environments. Section 4 presents the Fuzzy-PI drawbacks and the developped solution to improve the navigation performances and to optimize the Fuzzy-PI controller. An obstacle avoidance system based on fuzzy logic is detailed in section 5. Section 6 presents the real-time implementation of intelligent navigation system on Robotino PC-104 processor. In this section, experimental results are also detailed and performances of mobile robot navigation are evaluated. Finally, section 7 draws the conclusion.

2. Modeling and Kinematic control

Typically, robots, used in academic research or in industrial applications, are unicycles or car-like robot types. They have two degrees of freedom with two motion motors, but they didn't allow lateral movements. However, omnidirectional robots have three degrees of freedom with a minimum of three motors, making possible the side movements. Omnidirectional robots have the advantage of being flexible and agile, but, on the other hand, they are difficult to model and to control. They are stable and could move and rotate in any directions depending on the velocity of each wheel. Combining the movement of all three wheels causes forward, backward, sideways movements or rotation of the robot. Due to foregoing reasons, omnidirectional robots and vehicles are actually increasingly used in several academic and industrial projects [13] [14].

In this paper, we used an omnidirectionnal mobile robot, called Robotino [15], as a vehicular setup to simulate and experiment the proposed navigation algorithms. As illustrated in Fig.1, Robotino is equipped with three DC motors, webcam, nine infrared sensors use for distance measurements, three optical encoders to provide the actual position and speeds, binary collision detector, and WLAN interface board.

The mathematical development of the kinematic model will be performed through the relations between wheel and robot velocities.

Fig. 2 illustrates the localization and the orientation of the omnidirectional robot represented as a vector $(x, y, \phi)^T$. The global velocity of the robot is represented by the vector $(\dot{x}, \dot{y}, \dot{\phi})^T$ and the angular velocities of each wheel are represented by the vector $(\dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3)^{\mathrm{T}}$.

According to the model reported in [14], the Robotino kinematic model is given by:

$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} = \frac{1}{r}.\begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} = \frac{1}{r}.\overbrace{\begin{bmatrix} -\sin(\phi+\varphi_1) & \cos(\phi+\varphi_1) & R \\ -\sin(\phi+\varphi_2) & \cos(\phi+\varphi_2) & R \\ -\sin(\phi+\varphi_3) & \cos(\phi+\varphi_3) & R \end{bmatrix}}^{P(\phi)}.\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} \qquad (1)$$

With:

$\dot{\theta}_i$:	Angular speed of the wheel i
V_i:	Linear speed of the wheel i
r:	Radius of the wheel
R:	Distance between a wheel and the center of the robot
$\dot{\phi}$:	Angular speed of the robot
φ_i:	Angular location of the wheel i
$P(\phi)$:	Transformation matrix between the angular speeds of the wheels and the global velocity vector $(\dot{x}, \dot{y}, \dot{\phi})^T$.
i:	index of the wheel (i=1,2, 3)

The robot velocities could be determined from the velocity wheels by inverting the matrix $P(\phi)$ as follows:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = r.P^{-1}(\phi).\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} \qquad (2)$$

2.1 Kinematic control

Using the kinematic control, one could estimate the required speeds $(\dot{x},\ \dot{y},\dot{\phi})^T$ needed to navigate the omnidirectional robot from the current position coordinates $(x, y, \phi)^T$ to the target position coordinates $(x_d, y_d, \phi_d)^T$.
The difference between actual and final positions is given by :

$$\begin{bmatrix} x_e \\ y_e \\ \phi_e \end{bmatrix} = \begin{bmatrix} x \\ y \\ \phi \end{bmatrix} - \begin{bmatrix} x_d \\ y_d \\ \phi_d \end{bmatrix} \qquad (3)$$

The derivative of Eq. (3) gives the following relation:

$$\begin{bmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{\phi}_e \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = P^{-1}(\phi).\begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} \qquad (4)$$

To control the position of the mobile robot, we have to design the stabilizer and find the controlled wheels velocity vector to navigate Robotino from any starting position to any final destination. To stabilize the robot, we propose the following control law:

$$\begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} = P(\phi).\left(-K_p.\begin{bmatrix} x_e \\ y_e \\ \phi_e \end{bmatrix} - K_i.\begin{bmatrix} \int x_e \\ \int y_e \\ \int \phi_e \end{bmatrix} \right) \qquad (5)$$

where K_p and K_i are symmetric and positive definite coefficients. Hence, according to the error position system, the robot speed will take the following form:

$$\begin{bmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{\phi}_e \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} V_x \\ V_y \\ \Omega \end{bmatrix} = -K_p.\begin{bmatrix} x_e \\ y_e \\ \phi_e \end{bmatrix} - K_i.\begin{bmatrix} \int x_e \\ \int y_e \\ \int \phi_e \end{bmatrix} \qquad (6)$$

3. Fuzzy adaptive PI for Robotino navigation system

The main objective of the robot navigation is to reach the target position. Several parameters could affect the navigation steps such as the desired position and angle values, the traveled distances, the path curvature, the robot to target angle, the controller parameters (KP, KI) values as well as the linear and angular speeds. We start by designing a PI navigation controller to assist Robotino to reach the final position with a desired final angle. Experimental tests have shown that at anytime if we change the desired position, the robot reaches the target with an acceptable error depending on the robot final position and the desired angle. The precision of the conventional controller could be increased by adjusting the PI parameters (KP, KI) at each change of the desired position. This proves that PI parameters are not optimally chosen to obtain better and efficient results. To improve Robotino navigation performances, we develop a Mamdani-type fuzzy adaptive PI controller that combines the advantages of conventional PI with the fuzzy logic theory. Mamdani model is expressive and interpretable, the consequence of rules are fuzzy and more simple and intuitive. Fig. 3 shows the structure of the proposed control loop.

The fuzzy logic adaptive tuner has two inputs, the reference navigation error e and its variation ed, and two outputs, ΔK_p and ΔK_I . These two latter parameters are respectively the proportional and the integral gains variations. Input and output variables are defined by seven membership functions NB, NM NS, Z , PS, PM and PB as shown in figures 4, 5 , 6 and 7 respectively.

Relying on human knowledge, one could define a set of 98 linguistic rules summarised in the inference Table 1. These rules are obtained from the knowledge's of the relationship between navigation error e and its variation ed and PI parameters variation ΔKp and ΔKI. Each case of table 1 contains a logic rule and should be read as:
If (e is Ai) and (ed is Bj) then (ΔKp is Xij)(ΔKI is Yij)
where *Ai* , *Bj* , *Xij* , *Yij* are respectively the fuzzy membership functions of e, ed, ΔKp and ΔKI.

After defuzzification, new parameters of the PI controllers could be computed as follow:

$$K_p = K_{p0} + \Delta K_p \qquad (7)$$

$$K_i = K_{i0} + \Delta K_i \qquad (8)$$

where K_{p0} *and* K_{i0} are the PI controller original parameters.
The reference navigation error e is given by Eq. (3). PI discrete control law $u_i(k)$ of the wheel i could be then defined as follows:

$$u_i(k) = K_{Pi}.e_i(k) + K_{Ii}.T.\sum_{j=1}^{k} e_i(j) \qquad (9)$$

where k denotes the current discrete instant time and T is the sampling period.
In fact, the PI controller parameters $K_p = (K_{px}, K_{py}, K_{p\Omega})$ and $K_i = (K_{ix}, K_{iy}, K_{i\Omega})$ will be computed on-line by the fuzzy logic adaptive tuner (FLAT). Then, the PI control laws will be updated for the three speeds $(V_x, V_y, \Omega)^T$ to navigate the robot. The final step is to check the stop criterion depending on the collision detector value, the final position and the desired final angle. Hence, the intelligent navigation algorithm can be decomposed of the following steps:
Step 1. Initialization:

- Set the sample counter k=0;
- Establish the communication;
- Set the desired position and angle.

Step 2.

- Read the odometers outputs;
- calculate the position and orientation errors and their derivatives.

Step 3. FLAT:

- Compute on-line the fuzzification, the inference and the defuzzification steps,
- Provide values of parameters variation ΔK_p and ΔK_I.

Step 4.

- Update the gains K_P and K_I then the speeds V_x, V_y, Ω and finally the positions x, y, ϕ .

Step 5.

- Check the stop criterion. If the stop criterion is not reached, go to Step 2 and set the new value of the counter k=k+1 .

3.1 Simulation results of fuzzy-PI controller

Robotino SIM, which is an environment test tool provided by Festo Didactic [15], is used to simulate the omnidirectional robot navigation in a specific area (obstacles, lines, etc ...) before validating the experimental tests on Robotino. Based on this environment, Robotino has been programmed and

controlled using MATLAB platform to simulate the Fuzzy-PI navigation algorithm. Simulation tests have shown that, at anytime if we change the goal position, Robotino reaches the final position and the desired angle depending on the criterion values, as shown in figure 8. The holonomic robot has to move from the starting position $(x_0, y_0, \phi_0)^T$ to the target position of the path $(x_d, y_d, \phi_d)^T$. As shown in Fig. 9, Robotino can navigate in the four quadrants with different final positions; the final angle target is 10°. Moreover, five final orientation angles 10° , 20°, 30°, 40° and 60° are chosen (Fig. 10 and Fig. 11). Fig. 12 presents the output velocities of the proposed fuzzy-PI controller to achieve the trajectory tracking.
Navigation results confirm that Robotino is able to reach the target based on the fuzzy-PI controller, with a desired final distance and angle criterion. However, the proposed navigation algorithm presents two limitations: the presence of an error on the final robot angle as well as an error on the travelled distance length (Table 2 and Fig.10). In fact, the path curvature depends on Robotino final angle (Table 2 and Fig.11).

4. Fuzzy adaptive PI controller optimization

Previous results have shown that the Fuzzy-PI controller didn't perform well if the desired final angle increased. In this case, the robot performed a larger path with one degree error at the desired final angle. Experimental studies have shown that the error in the desired final angle increased for lower as well as for higher angle values. Deepen study of the navigation system have shown that the same fuzzy-controller could not be used to control together the linear and angular velocities. To overcome these limitations, we propose to replace the previous intelligent navigation system by two independent Mamdani fuzzy controllers. Indeed we have maintained the previous Mamdani-type fuzzy adaptive PI controller only for the control of linear velocities Vx and Vy, and designed a second Mamdani-type fuzzy system to control Robotino angular velocity as illustrated in figure 13.
The fuzzy logic adaptive tuner for the angular speed has three inputs, i.e. reference navigation error e, its variation de/dt and the distance error d; and two outputs $K_{p\Omega}$ and $K_{i\Omega}$. For the input variables e and de/dt and the output variables of the Mamdani fuzzy controller, seven membership functions are used, i.e. NB, NM NS, Z , PS, PM and PB as shown in figures 14, 15, 16 and 17 respectively. However, for the fuzzy input variable d, which defines the distance between the robot and its goal -Eq. (10), three membership functions VS, M and VB are used (Fig. 18).

$$d = \sqrt{(x_d - x)^2 + (y_d - y)^2} \qquad (10)$$

The controller was a Mamdani-type fuzzy model defined with a set of 147 linguistic rules where the i*th* rule is described by the following form:

Rule Ri : IF (e is Ai) And (de/dt is Bi) And (d is Ci) THEN ($K_{p\Omega}$ is Xi), and ($K_{i\Omega}$ is Yi),
where:

i :	the rule number
A_i:	ith membership function of the input e
B_i:	ith membership function of the input $\frac{de}{dt}$
C_i:	ith membership function of the input d
X_i:	ith membership function of the output $K_{p\Omega}$
Y_i:	ith membership function of the output $K_{i\Omega}$

4.1 Simulation results of fuzzy adaptive PI controller

The simulations are performed on the same platform as the previous Fuzzy-PI controller, i.e. Matlab for programming and Robotino Sim for GUI test. The robot had to reach the final position with a predefined orientation angle. The optimized fuzzy-controller aims to reduce the traveled distance and to reach more accurately the desired position compared to the previous controller. These objectives represent a challenge and constitute the major contribution of this work. In fact, the basic improvement of the proposed control scheme can be highlighted in two points: (i) the optimized fuzzy controller must mainly reduce the Robotino deviation at the beginning of the navigation phase; this automatically leads to the reduction of path curvature and traveled distance and (ii) limitation of the error on the desired destination. This is the main objective of the added angular velocity Fuzzy-PI controller. Navigation results based on the optimized fuzzy-controller, show that Robotino was able to reach the target with the desired final position and angle. Experimental results show great improvements of Robotino navigation performances compared to single Fuzzy-PI controller. In fact, the traveled distance as well as the error on the desired direction angle (greater than 45°) have been decreased significantly (see Table 3 and Fig.20).

5. Obstacle avoidance

5.1 Design of Mamdani-type fuzzy logic obstacle avoidance controller

The obstacle avoidance process is a crucial step during Robotino navigation phase. Initially, we develop a classic If-Then avoidance algorithm, which has the disadvantages of giving brutal speed changes when it passes from one loop to another. In addition, it does not cover all possible cases of obstacle locations. Using the human-based knowledge experience, a fuzzy-logic obstacle avoidance controller could be used to overcome these two disadvantages. In fact, it could cover the majority of possible cases of obstacle locations as well as a smoothly change of the robot speed. To fulfill this task, we have developed a third Mamdani-type fuzzy controller.

In this section, we will present an approach based on Mamdani-fuzzy controller that allows the omnidirectionnal mobile robot to avoid fixed obstacles in an unknown environment.

For the purpose of obstacles avoidance, the Robotino is equipped with 9 proximity infrared sensors which are used in real-time detection of obstacles by measuring the reflected signal. Infrared sensors are located on the front platform at 40° one each other from the central axis. Figure 21 shows the robot infrared sensors location.

For this study, only the robot frontal five infrared (IR) proximity sensors are used: IR1, IR2, IR3, IR8, IR9. In fact, Robotino will navigate to the target position while avoiding fixed obstacles in his way. For this purpose, a Mamdani-fuzzy logic technique is used in order to design a robust control law for the navigation in an environment with unknown obstacles. The main idea is to get the infrared sensor scope. If the voltage response of the IR is lower than 0.7 V, which corresponds to a distance of 18 cm, then the mobile robot can move towards the target, otherwise, an obstacle is detected. In this case, the third Mamdani-fuzzy logic controller is applied to allow the omnidirectional mobile robot to move away from the detected obstacle. Figure 22 describes the navigation principle with obstacles avoidance.

The Mamdani-fuzzy logic controller will get instantaneously the response of Robotino sensors (IR1, IR2, IR3, IR9, IR8) and will provide the velocity (Vxob and Vyob) in (x,y) plan as well as the angular velocity (Ωob). In this case, the three fuzzy sets (Far, Middle, Close) are used to describe the distance information between Robotino and detected obstacle converted into voltage. Membership functions are presented in figure 23.

The outputs of Mamdani-fuzzy avoidance controller are Vxob, Vyob and Ωob. The velocity Vxob is described using three fuzzy sets (Zero, Forward, Forward Big). The velocity Vyob is described using five fuzzy sets (Backward Big, Backward, Zero, Forward, Forward Big) and finally five fuzzy sets are used for the angular velocity Ωob (Negative Big, Negative, Zero, Positive, Positive Big). Membership functions of these outputs are presented in figures 24, 25 and 26 respectively.

In order to avoid unknown obstacles, the fuzzy controller operates with 25 rules. These rules are obtained from the knowledge's of the relationship between sensors values and both of the linear and the angular velocities. Some rule-bases are described as follow:
If (IR1 is Close) and (IR2 is Far) and (IR9 is Far) then (Vxob is Zero)(Vyob is Forward Big)(Ωob is Positive Big)
If (IR1 is Far) and (IR9 is Close) and (IR8 is Far) then (Vxob is Forward Big)(Vyob is Forward Big)(Ωob is Positive Big)
If (IR1 is Far) and (IR9 is Close) and (IR8 is Middle) then (Vxob is Forward)(Vyob is Forward Big)(Ωob is Positive Big)
If (IR1 is Middle) and (IR2 is Close) and (IR3 is Far) then (Vxob is Forward)(Vyob is Backward)(Ωob is Positive)
If (IR1 is Far) and (IR9 is Middle) and (IR8 is Close) then (Vxob is Forward)(Vyob is Zero)(Ωob is Positive Big)
If (IR1 is Close) and (IR2 is Close) and (IR9 is Middle) and (IR3 is Middle) and (IR8 is Far) then (Vxob is Forward Big)(Vyob is Backward Big)(Ωob is Negative Big)
If (IR1 is Close) and (IR2 is Close) and (IR9 is Close) and (IR3 is Middle) and (IR8 is Middle) then (Vxob is Zero)(Vyob is Forward Big)(Ωob is Positive Big)
If (IR1 is Far) and (IR2 is Middle) and (IR9 is Far) and (IR3 is Middle) then (Vxob is Forward Big)(Vyob is Backward Big)(Ωob is Negative Big)

⋮

⋮

If (IR2 is Close) and (IR3 is Close) then (Vxob is Forward)(Vyob is Zero)(Ωob is Negative Big)
If (IR9 is Far) and (IR8 is Close) then (Vxob is Forward)(Vyob is Forward)(Ωob is Positive Big)
The Mamdani-fuzzy obstacles avoidance controller outputs are computed using centroid deffuzification method and and are expressed by the following equations:

$$\mathrm{Vxob} = \frac{\sum_{i=1}^{r} \alpha_i x_i}{\sum_{j=1}^{r} \alpha_j} \qquad (11)$$

$$\mathrm{Vyob} = \frac{\sum_{i=1}^{r} \beta_i y_i}{\sum_{j=1}^{r} \beta_j} \qquad (12)$$

$$\Omega\mathrm{ob} = \frac{\sum_{i=1}^{r} \gamma_i \varphi_i}{\sum_{j=1}^{r} \gamma_j} \qquad (13)$$

where α_i, β_i and γ_i are the level activation of the rule i.

5.2 Simulation results of obstacles avoidance system

Simulation results are presented firstly using one obstacle and then using two obstacles. In fact, Robotino navigates in an unstructured environment with unknown obstacles. Figure 27 describes simulation results with a left obstacle realized using Robotino sim. Figures 28, 29 and 30 present the corresponding results with Robotino Sim and Matlab environments. In this case, the target position is (2000mm, 1500mm) and the desired final angle equals 40°. Results show that when Robotino detects a left obstacle with IR2 and IR3 sensors (Figure 29), it will automatically generate the corresponding speeds (Figure 30) using its fuzzy obstacles avoidance controller. The position and the angle of the robot (Figure 28) are then affected by speed variations. When there are no obstacles on the robot path, it resumes its navigation algorithm.

Figure 31 shows the navigation sequential images realized for the case of two obstacles: the first one is at the right side of the robot and the second one is at the left side. Figures 31-34 present simulation results with Robotino Sim and Matlab environments. In this case, the target position is (2500mm, 2000mm) and the desired final direction angle equals 30°. The obtained results show that when Robotino detects the first right obstacle with IR9 and IR8 sensors, then the second left obstacle with IR2 and IR3 (Figure 33). Based on these values, the fuzzy obstacles avoidance controller will generate the corresponding speeds (Figure 34). The position and the angle of the robot (Figure 32) are then generated based on speeds variations. When there are no obstacles detected on the robot path, the latter resumes its navigation algorithm.

6. Experimental results

Robotino is used as a vehicular setup to experiment our navigation and obstacle avoidance algorithms. Robotino is a mobile robotic system operating with high quality omnidirectional drive. This holonomic mobile robot is provided with a webcam and several sensor types like infrared analog measuring distances, binary collision bumper and moved by three DC motors equipped with optical encoders to control the actual position/speed. Its PC-104 board processor includes a 10/100 Mbit Ethernet port, 2 USB interfaces, VGA connector, mouse and keyboard interfaces, RS232, Wireless LAN interface board, a watchdog timer and real time clock. Robotino have a very advanced programming platform using Linux real-time OS. Indeed, it can be programmed with many languages like Robotino View, C++, JAVA, Visual Basic, Labview and Matlab/Simulink. The latter is used to implement intelligent navigation algorithms. To estimate the performances of the Mamdani-fuzzy controllers for navigation and obstacles avoidance, experimental tests are performed for two and three obstacles.

6.1 Robotino experimental test using two obstacles

To evaluate real performances of fuzzy controllers, when the robot navigates in an unstructured environment with unknown obstacles, experimental tests are presented using two obstacles.
Initially, the robot is supposed to not detect obstacles and will start moving with at the respectively speeds $(V_x, V_y, \Omega)^T$. As the IR9 and IR8 sensors start detecting the obstacle at the right side of Robotino (Fig 37-IR9-IR8), the obstacle avoidance algorithm activates the linear velocities Vx and Vy, and the speed will decrease considerably (Fig.38- Vx/Vy speed). The angular velocity will generate negative values of angular speed (Fig.38) to avoid obstacles as illustrated in the pictures 4-7 of Figure 35. At the instant, corresponding to the iteration 75, IR2 and IR3 have detected an obstacle at the left side of Robotino (Fig 37-IR2-IR3). While the fuzzy obstacle avoidance algorithm still activated, Robotino will navigate at low velocities values of Vx and Vy (Fig.38- Vx/Vy speed). The angular velocity fuzzy avoidance controller will generate adequate values to avoid the obstacles (Fig.38- Ang speed) (Fig 35-Pictures 8-15). At iteration 150, there are no obstacles on the robot path and the robot resumes its navigation algorithm (Fig 35-Pictures 16-24). At the time corresponding to iteration 2158 the robot reaches its target (Fig.36).

6.2 Robotino experimental test using three obstacles

To evaluate the Robotino skills to navigate and avoid more obstacles. We perform the experimental test using three obstacles. In this case, the target position is (2500mm, 2000mm) and the desired final direction angle equals 30°. The experimental test result using three obstacles are illustrated in the figures 39-42. The latter shown the successive pictures navigation, position and angle, infrared sensors voltage and speeds values. These experimental results have validated Robotino navigation algorithms in an unstructured environment with unknown obstacles.

7. Conclusion

The present work details the design, optimization, simulation and experimental tests of fuzzy-PI controllers for an omnidirectional robot navigating system. The main purpose of the work is to navigate Robotino to reach the target in response to different formulated requirements: (i) reach the target position with a desired final angle, (ii) minimize the traveled distance and (iii) avoid occurred obstacles. To achieve these requirements, a kinematic model, detailing the Robotino system, was given and a navigation Fuzzy-PI controller has been developed. Simulation results, using MATLAB and Robotino-Sim platform, have shown that Robotino was able to reach the target based on the Fuzzy-PI controller, with a desired goal position and orientation angle. However two limitations have been raised: an error on the final robot angle and a much longer travelled distance. To overcome these limitations, we have proposed a solution based on the replacement of the previous intelligent navigation system by two independent controllers. Indeed, we have maintained the previous controller to control the robot linear velocities and we have designed a second fuzzy system to control the robot angular velocity. Adaptive Fuzzy-PI controllers adjust their parameters to decrease the error caused by the dynamics changes and navigation challenges of Robotino. To achieve the navigation task, we have developed an obstacle avoidance system based on fuzzy logic. Hence, using the proposed navigation system based on three fuzzy controllers, simulation and experimental results have shown a clear improvement of the navigation system performances. Thus, the fuzzy implemented controllers have shown its adaptation to the dynamics of real omnidirectional robots. In fact, Robotino was able to reach the target position with a desired final angle direction and to avoid obstacles based on the intelligent controllers. Moreover, the proposed intelligent controllers can be applied to other nonlinear dynamics process such as ITS or omnidirectional wheelchair system.

References

[1] Rossomando F. G., Soria C. M.: Identification and control of nonlinear dynamics of a mobile robot in discrete time using an adaptive technique based on neural PID. Int. J. Neural Comput & Applic, pp: 1179 - 1191, 2015.

[2] Ching-chih T., Xiao-ci W. A., Feng-chvn T. And Chun-chieh C.: Fuzzy decentralized elf-based pose tracking for autonomous omnidirectional mobile robot, 2014 International Conference on Machine Learning and Cybernetics, Lanzhou, China, pp: 748 - 754, 2014.

[3] Hernandez D.C., Van-Dung H. ; Filonenko A. and Kang-Hyun Jo.: vision-based heading angle estimation for an autonomous mobile robots navigation. IEEE 23rd International Symposium on Industrial Electronics (ISIE), Istanbul, Turkey, pp: 1967-1972, 2014.

[4] Peña M., Gómez J. A., Osorio R., López I., Lomas V., Gómez H. and Lefranc G.: Fuzzy Logic for Omnidirectional Mobile Platform Control Displacement using FPGA and Bluetooth Communication Devices. International. IEEE latin america transactions, VOL. 13, NO. 6, JUNE 2015, pp: 1907-1914, JUNE 2015.

[5] Klabi I., Benjemmaa A., Masmoudi M.S. and Masmoudi, M.: Computer Optimum Architecture of Neural Networks lane following system. International Journal of Computer Applications Volume 42– No.12, pp: 41-46, 2012.

[6] Daji Tian, Shaoping W. and El Kamel A. : Fuzzy controlled avoidance for a mobile robot in a transportation optimization. International Conference on Fluid Power and Mechatronics (FPM), China, pp: 868 - 872, 2011.

[7] Hsu-Chih H., Ter-Feng W., Huan-Shiuan H., Chih-Chiang Y. and Chien-Lung W.: Intelligent Motion Control for Omnidirectional Mobile Robots Using Artificial Immune System Algorithm. SICE Annual Conference 2013, September 14-17, Nagoya University, Nagoya, Japan, pp: 431-434, September 14-17, 2013.

[8] Treesatayapun C. and Guzman-Carballido A.: Linearization based on Fuzzy Rules Emulated Networks for nonaffine discrete-time systems controller. TENCON 2009 - IEEE Region 10 Conference, TENCOM, Singapore, pp:1-6, 2009.

[9] Weinert H. and Pensky, D. : Mobile Robotics in Education and Student Engineering Competitions, IEEE Africon Zambia, pp: 1-5, 2011.

[10] Oltean S. E., Dulău M. and Puskas R.: Position Control of Robotino Mobile Robot Using Fuzzy Logic. IEEE International Conference on Automation Quality and Testing Robotics (AQTR), Romania, pp:1-6, 2010.

[11] Sheikhlar A., Fakharian A. and Adhami-Mirhosseini A.: Fuzzy Adaptive PI Control of Omni-Directional Mobile Robot. 13th Iranian Conference on Fuzzy Systems (IFSC), Iran, pp:1-4, 2013.

[12] Hsu-Chih H., Ter-Feng W., Chun-Hao Y. and Huan-Shiuan H.: Intelligent Fuzzy Motion Control of Three-Wheeled Omnidirectional Mobile Robots for Trajectory Tracking and Stabilization. International Conference on Fuzzy Theory and Its Applications, Taiwan, 2012.
[13] Radovnikovich M., Fleck P., and Hallenbeck K., Ultra wide-band trajectory following for an omnidirectional factory automation vehicle, 2014 IEEE International Conference On Technologies for Practical Robot Applications (TePRA), pp:1-6, 2014.

[14] T.A. Baede, Motion Control of an Omnidirectional Mobile Robot, Traineeship report DCT, National University of   Singapore, Faculty of Engineering, Department of Mechanical Engineering, 2006.

[15] http://www.festo-didactic.com.`

Fuzzy Logic Controllers Design For Omnidirectionnal Mobile Robot Navigation

(Figures)

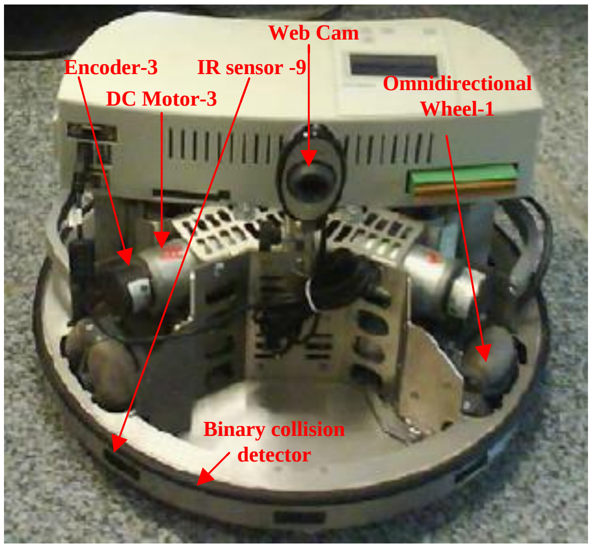


Figure 1. Robotino

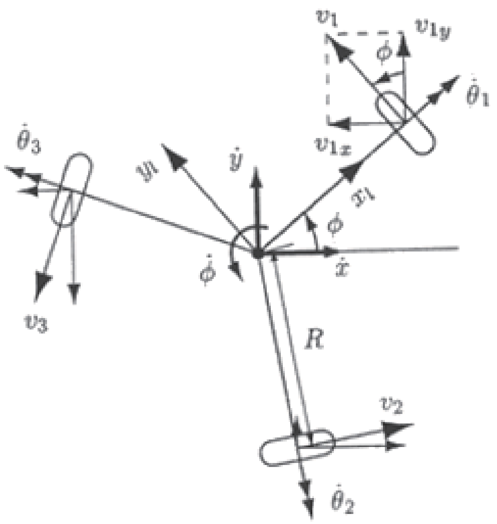


Figure 2. Kinematic diagram of the robot

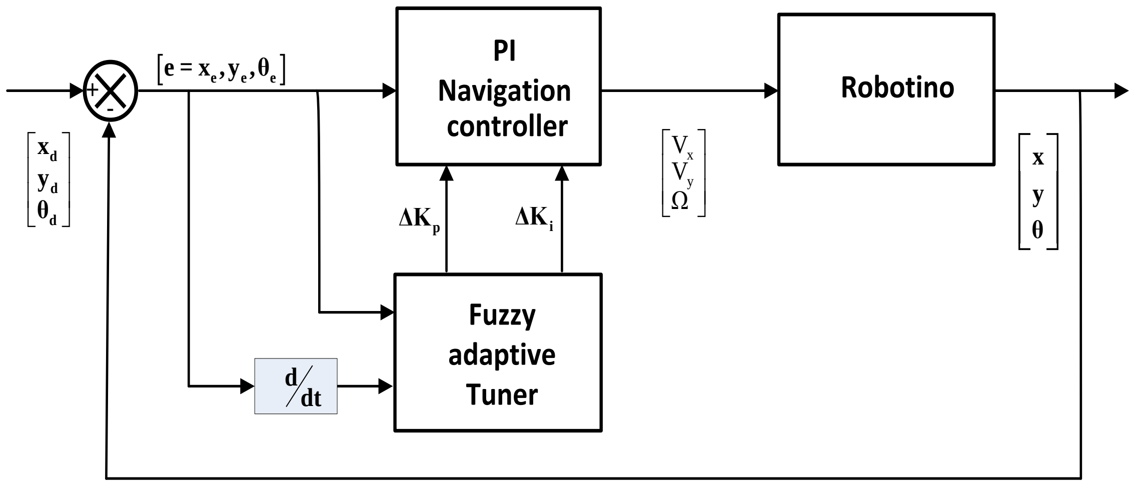

+
-
$[e = x_e, y_e, \theta_e]$
$\begin{bmatrix} x_d \\ y_d \\ \theta_d \end{bmatrix}$
PI Navigation controller
ΔK_p
ΔK_i
$\begin{bmatrix} V_x \\ V_y \\ \Omega \end{bmatrix}$
Robotino
$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$
d/dt
Fuzzy adaptive Tuner

Figure 3. Fuzzy adaptive PI control loop

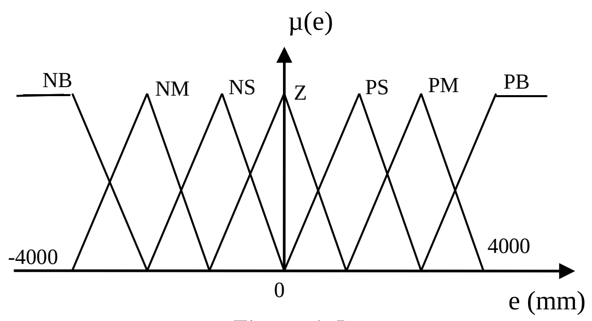


Figure 4. Input error

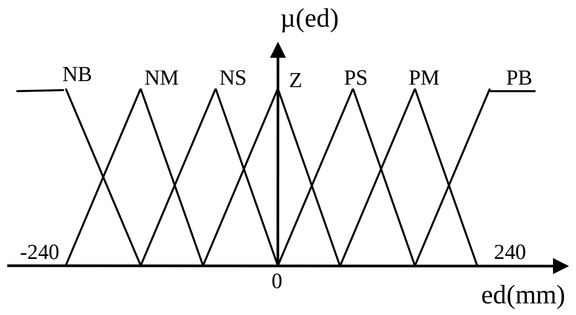


Figure 5. Input error variation

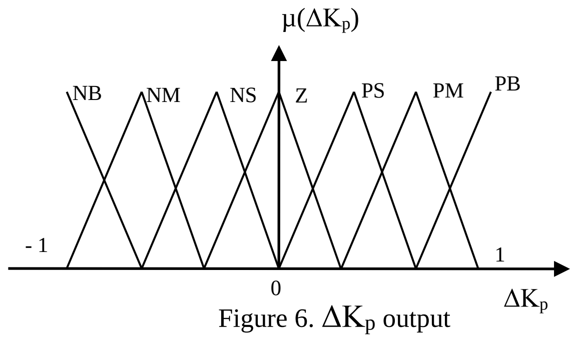


Figure 6. ΔK_p output

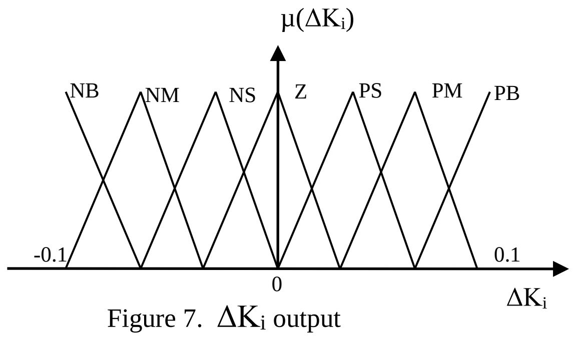


Figure 7. ΔK_i output

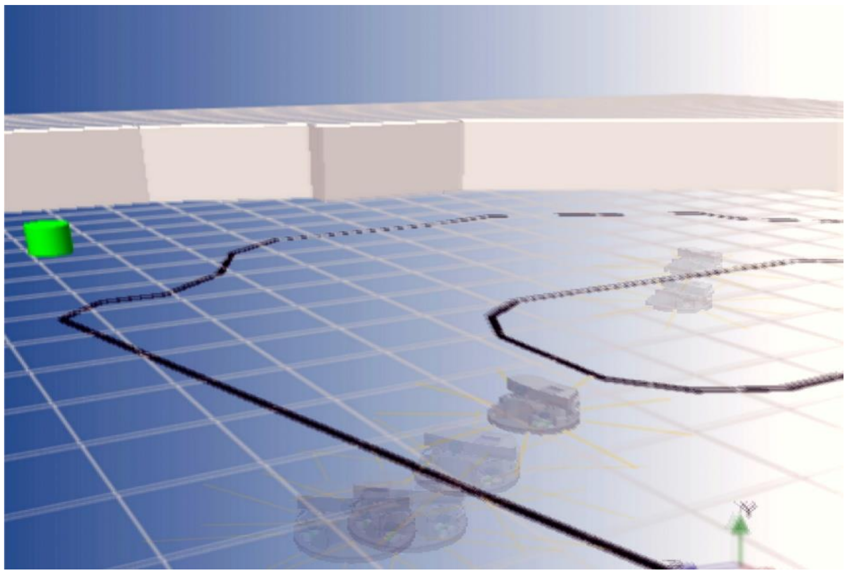


Figure 8. Robotino SIM trajectories: the target destination is (3500, 2000) and the final orientation angle is 30°

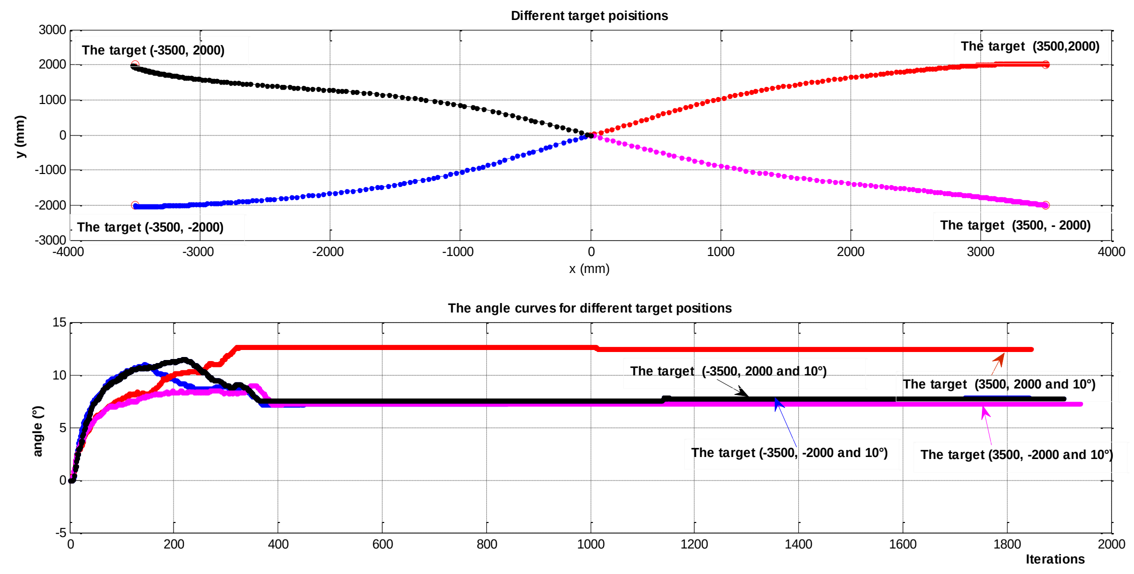


Figure 9. Trajectories for different target positions (± 3500, ± 2000) in the four quadrants-the final orientation angle is 10°

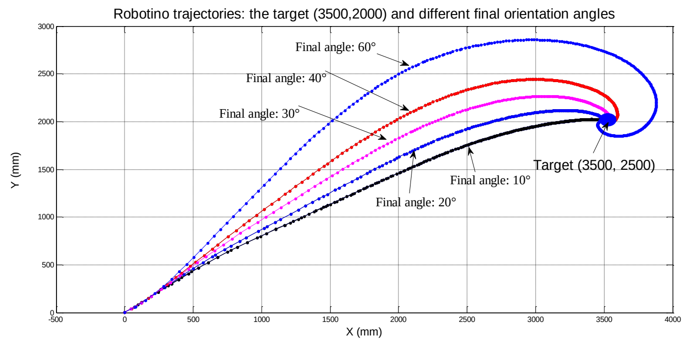


Figure 10. . Robotino trajectories with different final orientation angles (10°, 20°, 30°, 40° and 60°) and the target destination (3500, 2000)

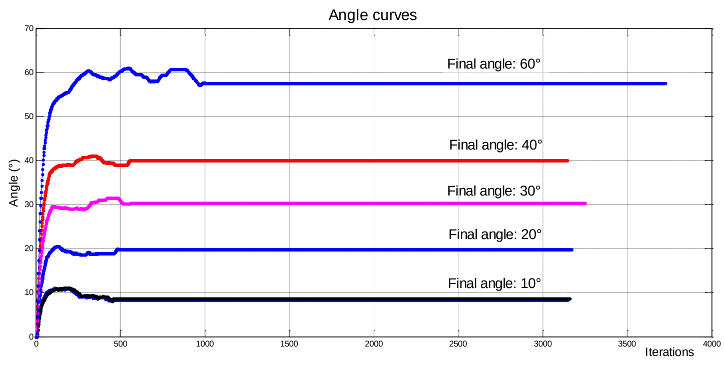


Figure 11. Robotino angle curves with different final orientation angles (10°, 20°, 30°, 40° and 60°), the target destination (3500, 2000)

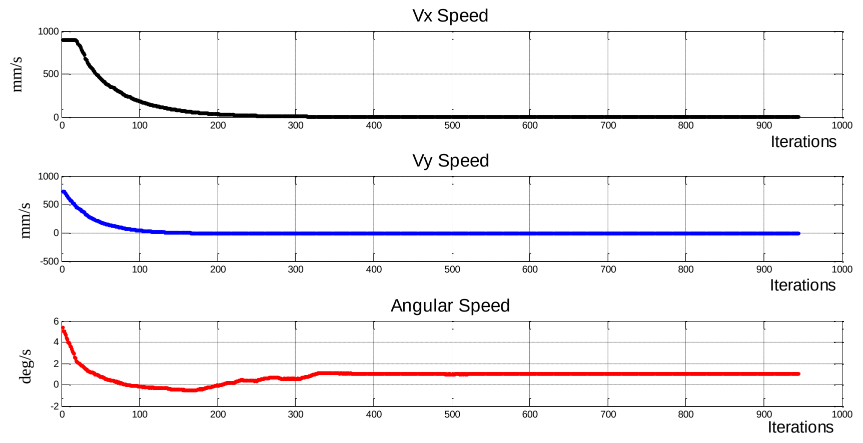
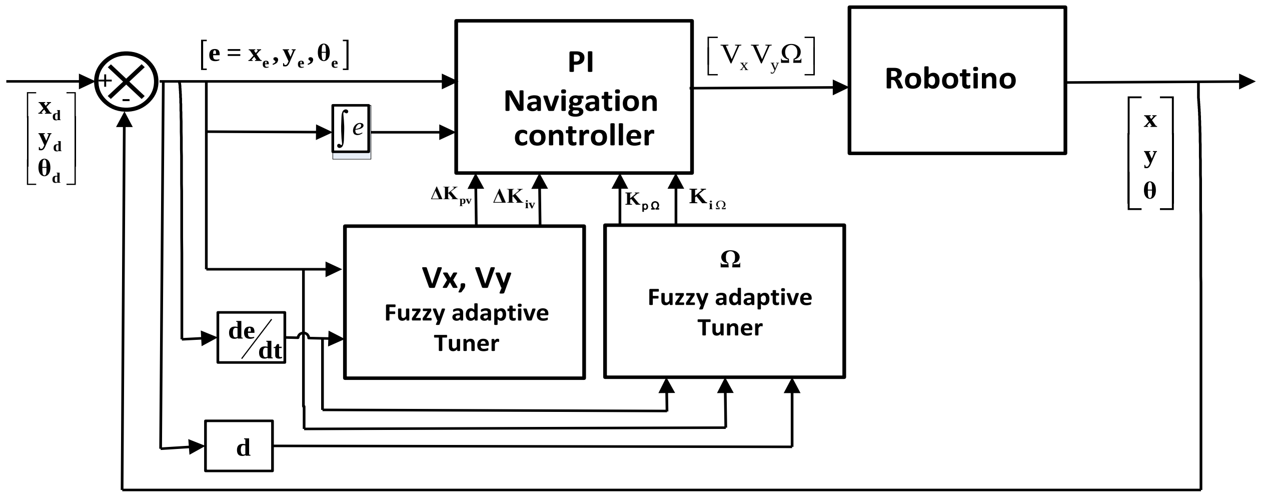


Figure 12. Speeds values to reach the final destination (3500, 2000), the final orientation angle is 10°



$[e = x_e, y_e, \theta_e]$
PI Navigation controller
$[V_x V_y \Omega]$
Robotino
x_d y_d θ_d
x y θ
ΔK_{pv} ΔK_{iv} $K_{p\Omega}$ $K_{i\Omega}$
Vx, Vy Fuzzy adaptive Tuner
Ω Fuzzy adaptive Tuner
de/dt
d

Figure 13. Fuzzy adaptive tuner for linear and angular speeds

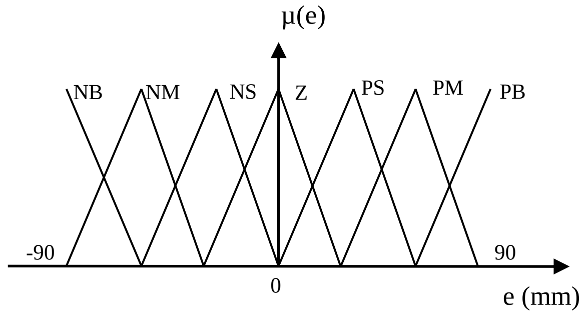


Figure 14. Angle error input

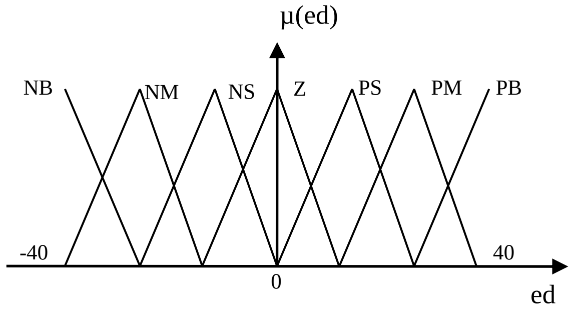


Figure 15. Angle error variation input

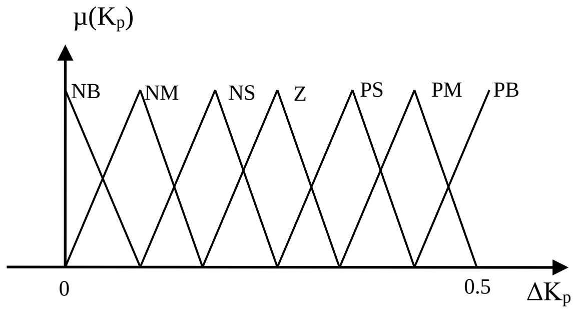


Figure 16. $K_{p\Omega}$ output

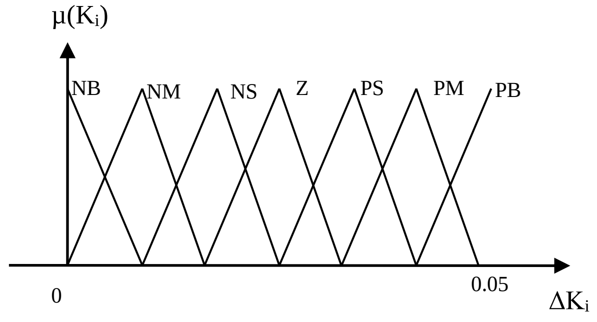


Figure 17. $K_{i\Omega}$ output

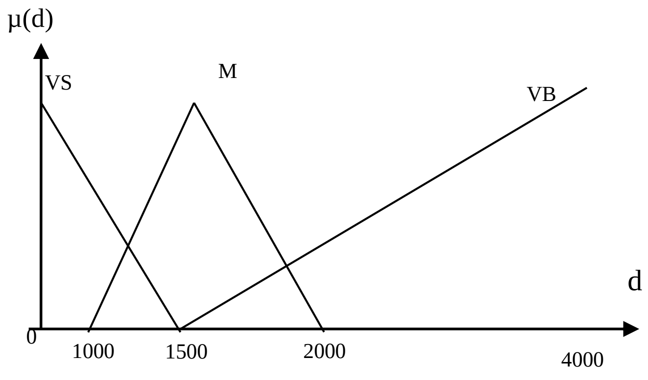


Figure 18. Distance error input

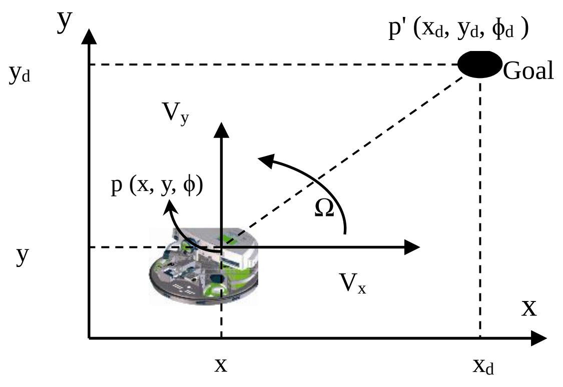


Figure 19. Robotino schematic model

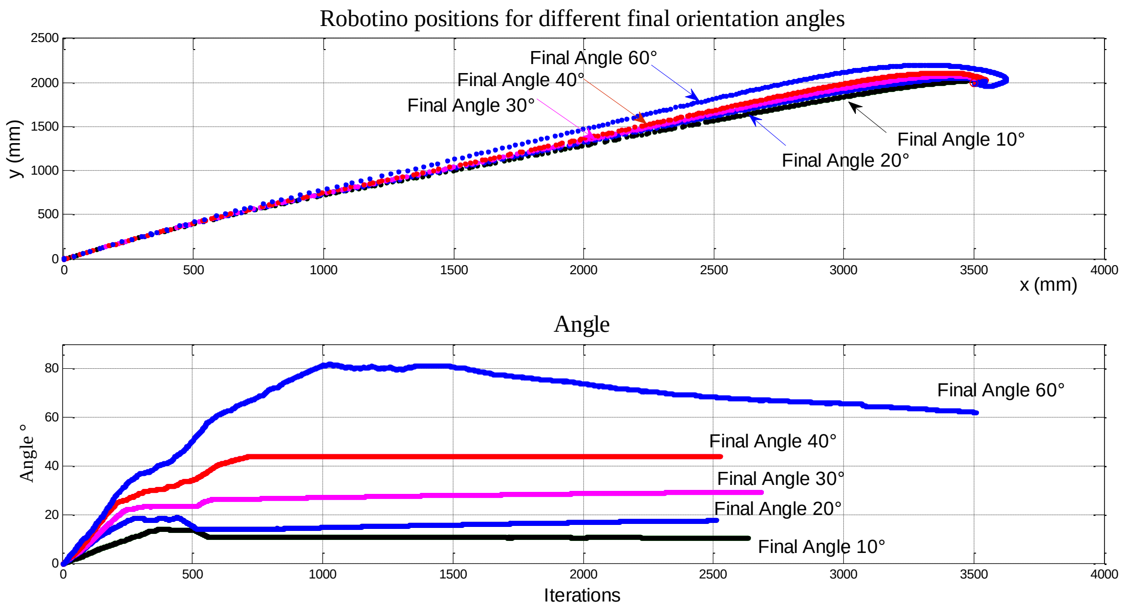


Figure 20. Robotino optimized trajectories: the target destination (3500, 2000) and different final orientation angles (10°, 20°, 30°, 40° and 60°)

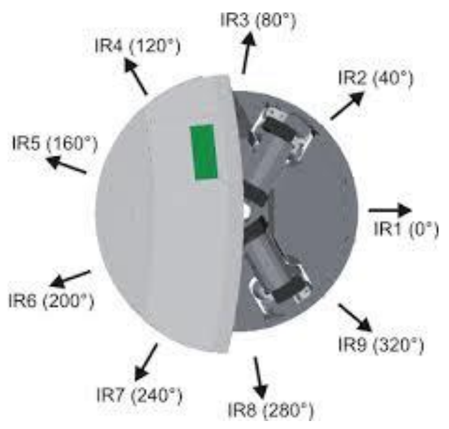


Figure 21. IR sensor positions.

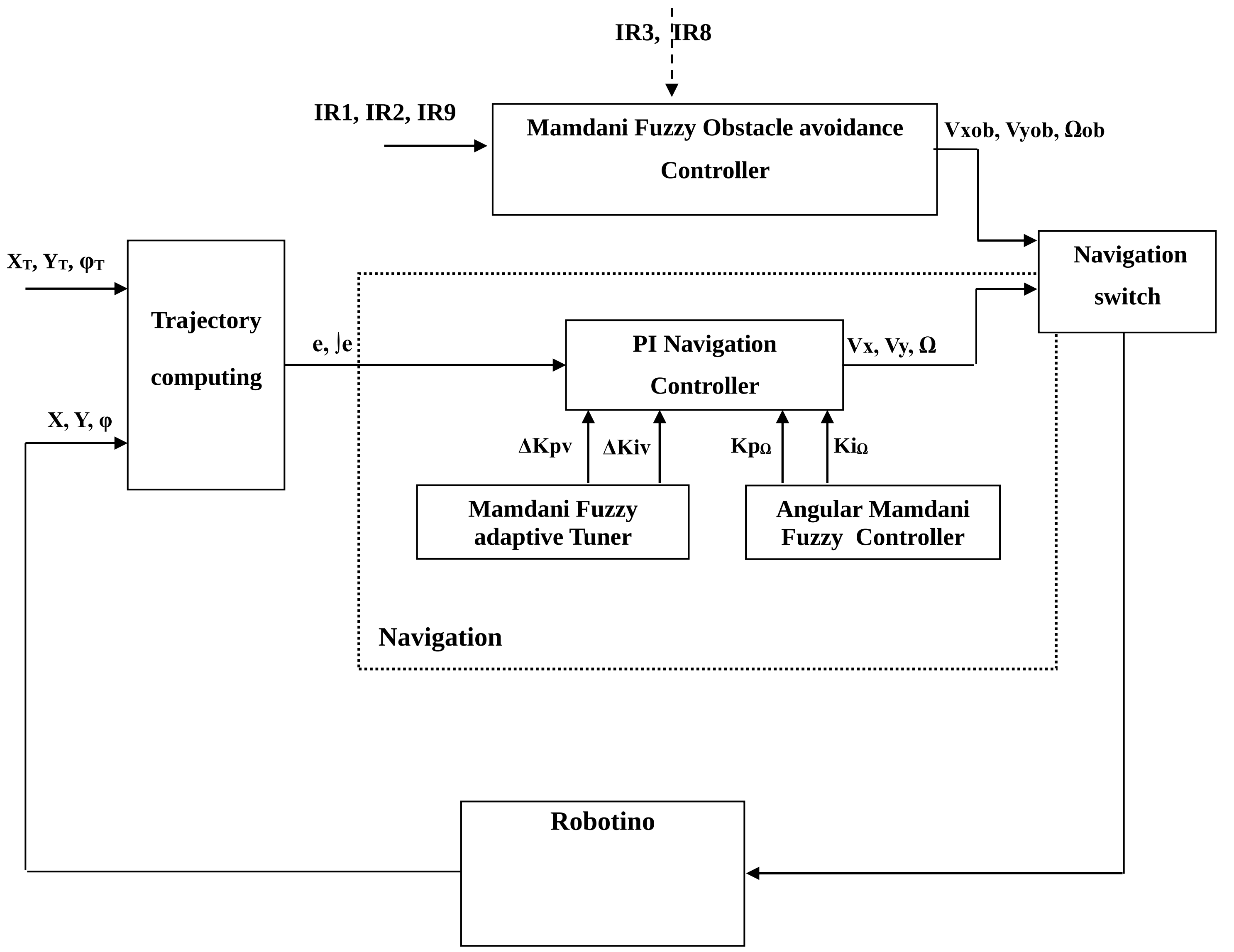


Figure 22. Mamdani Fuzzy avoidance obstacles controller

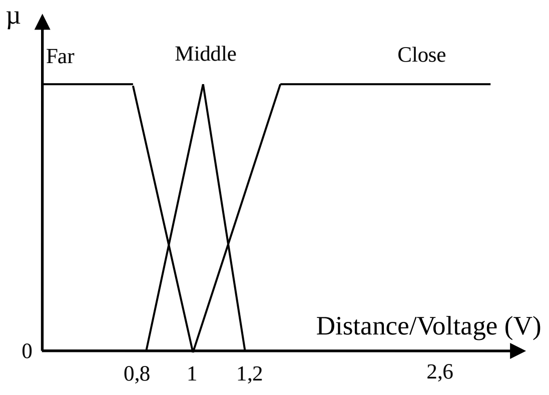


Figure 23. Membership function plots of IRi/i=1, 2, 3, 8, 9

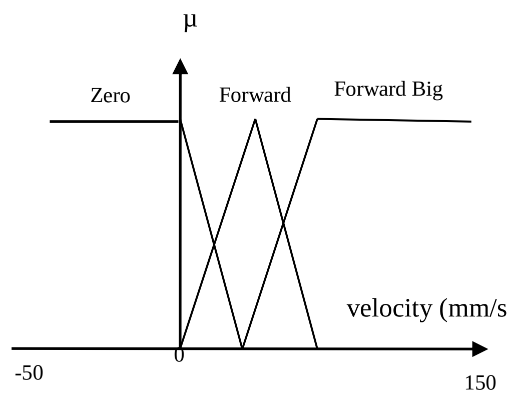


Figure 24. Membership function plots of Vxob

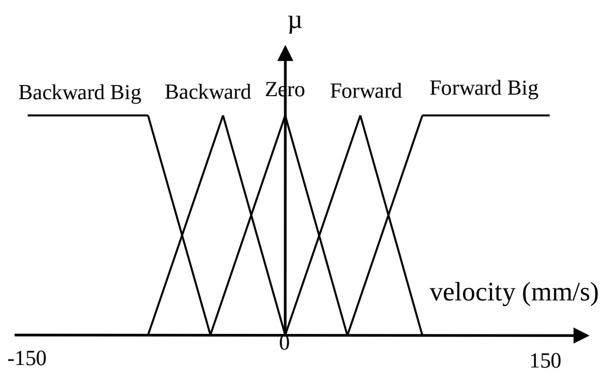


Figure 25. Membership function plots of Vyob

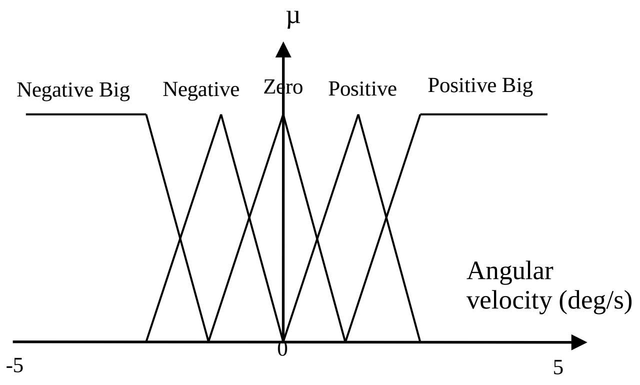


Figure 26. Membership function plots of angular velocity Ωob

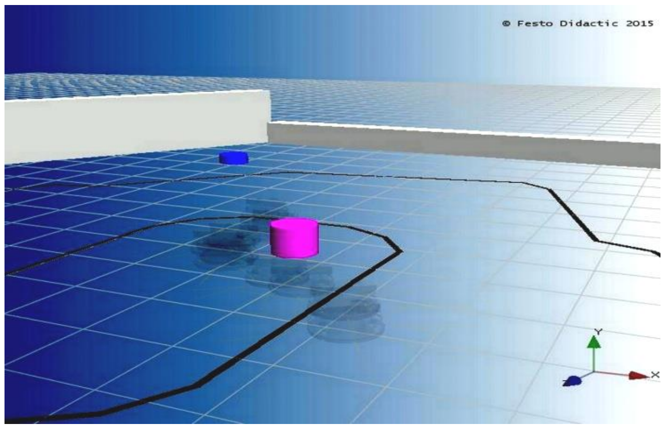


Figure 27. Robotino SIM trajectory with a left obstacle

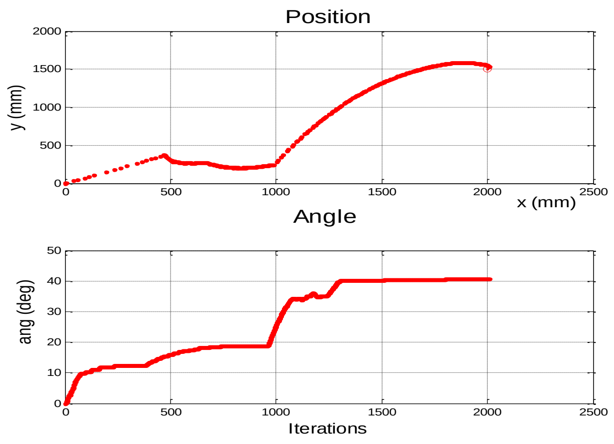


Figure 28. Left obstacle simulation result of position and angle: the target destination is (2000 mm, 1500 mm) and the desired orientation angle is 40°

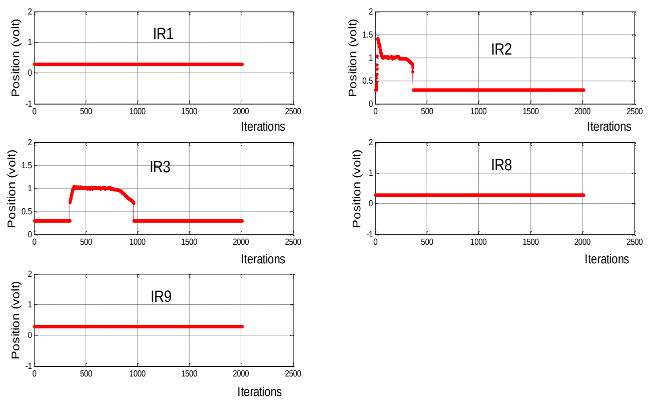


Figure 29. The Inputs infrared sensors values with a left obstacle

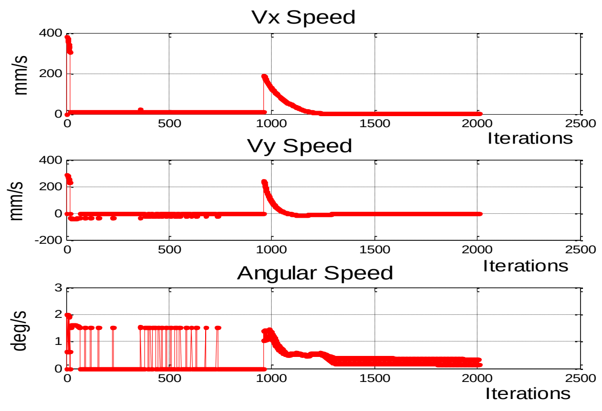


Figure 30. Speeds values to avoid left obstacle and to reach the final destination, the final orientation angle is 40°

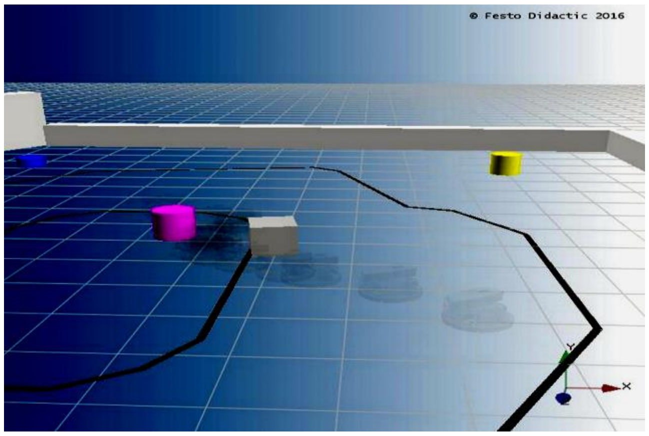


Figure 31. Robotino SIM trajectory with two obstacles

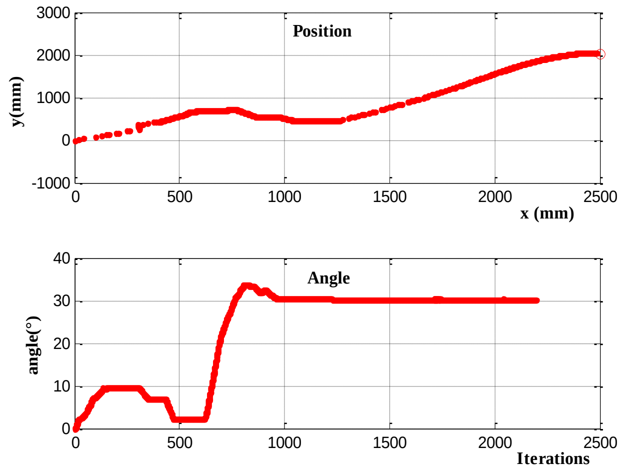


Figure 32. Simulation result of position and angle for two obstacles: the target destination is (2500, 2000) and the final orientation angle is 30°

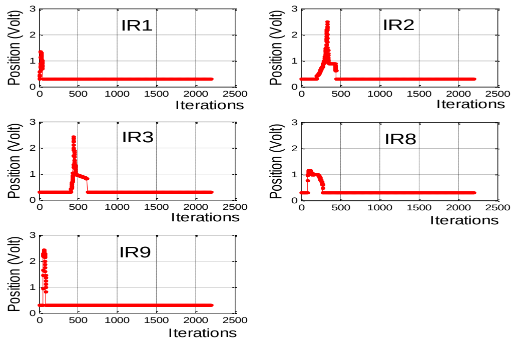


Figure 33. Input infrared sensors with two obstacles

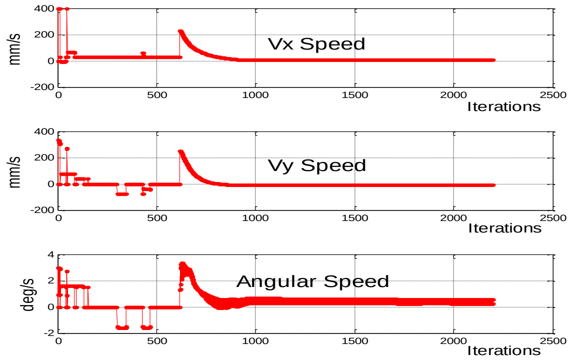


Figure 34. Speeds to avoid two obstacles and reaching the final destination, the desired orientation angle is 30°

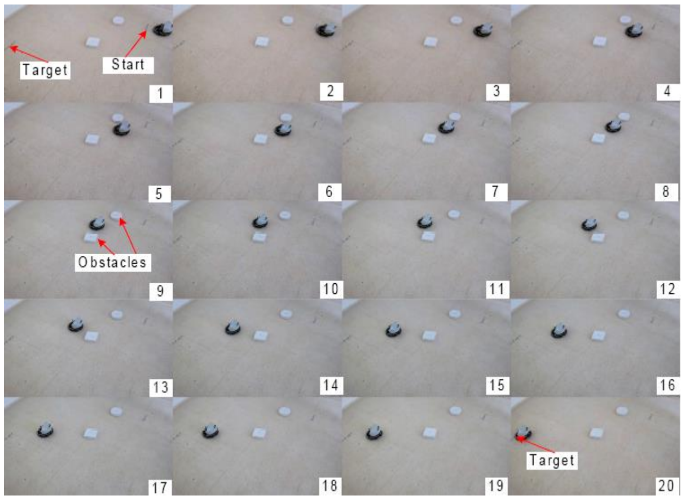


Figure 35. Successive pictures of navigation trajectory with two obstacles

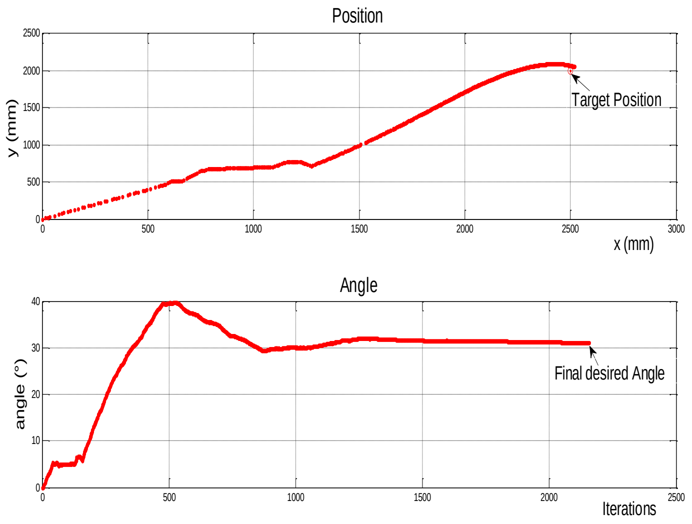


Figure 36. Two obstacles experimental result of position and angle: the target destination is (2500, 2000) and the desired orientation angle is 30°

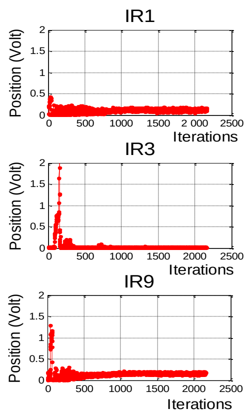


Figure 37. Inputs infrared sensors (experimental values) with two obstacles

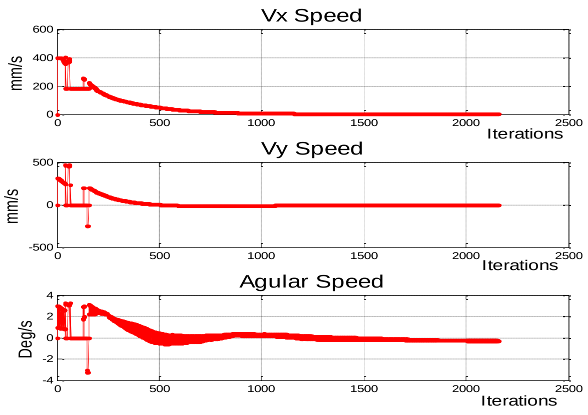


Figure 38. Speeds values $(V_x, V_y, \Omega)^T$ to avoid two obstacles to reach the final destination, the final desired orientation angle is 30°

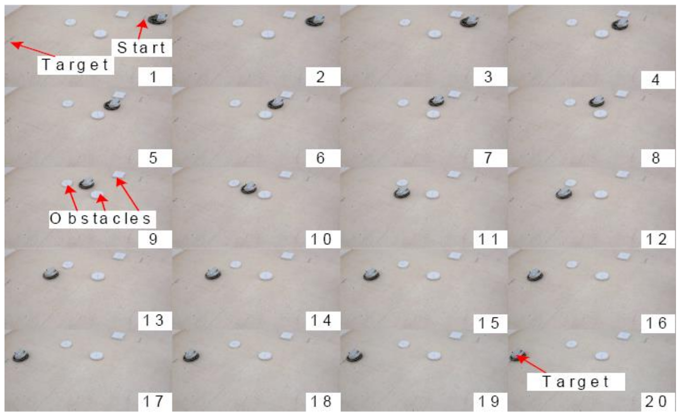


Figure 39. Successive pictures of navigation trajectory with three obstacles

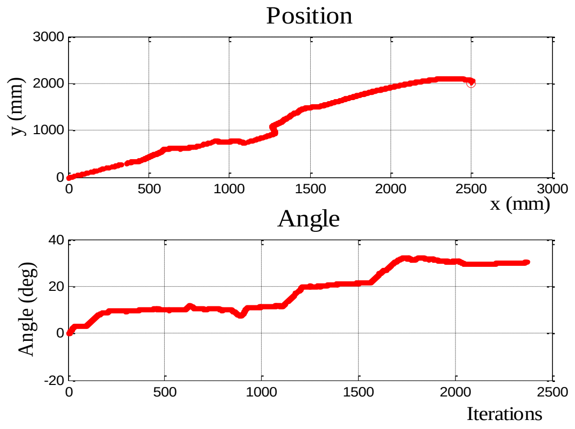


Figure 40. Three obstacles experimental result of position and angle: the target destination is (2500, 2000) and the desired orientation angle is 30°

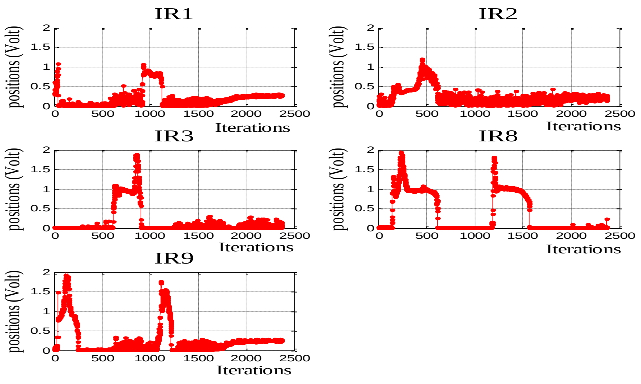


Figure 41. Inputs infrared sensors (experimental values) with three obstacles

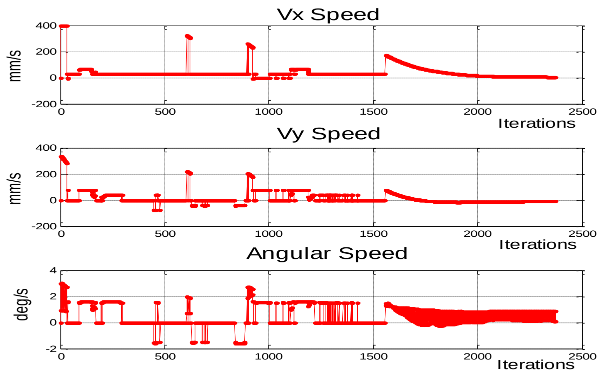


Figure 42. Speeds values $(V_x, V_y, \Omega)^T$ to avoid three obstacles to reach the final destination, the final desired orientation angle is 30°

Table1. Rule-base and output weights of the Fuzzy adaptive PI

		Error variation (ed)													
		NB2		**NM2**		**NS2**		**Z2**		**PS2**		**PM2**		**PB2**	
		ΔK_p	ΔK_i	ΔK_p	ΔK_i	ΔK_p	ΔK_i	ΔK_p	ΔK_i	ΔK_p	ΔK_i	ΔK_p	ΔK_i	ΔK_p	ΔK_i
Error (e)	**NB1**	PB	NB	PB	NB	PM	NM	PM	NM	PS	NS	Z	Z	Z	Z
	NM1	PB	NB	PB	NB	PM	NM	PS	NS	PS	NS	Z	Z	NS	Z
	NS1	PM	NB	PM	NM	PM	NS	PS	NS	PS	Z	Z	PS	NS	PS
	Z1	PM	NM	PM	NM	PS	NS	Z	Z	NS	PS	NM	PM	NM	PM
	PS1	PS	NM	PS	NS	Z	Z	NS	PS	NS	PS	NM	PM	NM	PB
	PM1	PS	Z	Z	Z	NS	PS	NM	PS	NM	PM	NM	PB	NB	PB
	PB1	Z	Z	Z	Z	NM	PS	NM	PM	NM	PM	NB	PB	NB	PB

Table 2: The travelled distance length related to the final robot angle for the target destination (3500, 2000)

Desired Angle °	0°	20°	40°	60°	80°
Robot Final Angle °	4.2	18.1	40.3	57.9	76.7
Ang. error °	-4.2	1.9	-0.3	2.1	3.3
Travelled Distance (m)	4.041	4.155	4.742	6.159	15.361
Dist. error %	0.2	3	17.6	55	281

Table 3: The optimized travelled distance length related to the final robot angle the target destination (3500, 2000)

Desired Angle °	0°	20°	40°	60°	80°
Robot Final Angle °	5.3	17.74	44.1	61.75	82
Angular error °	-5.3	2.26	-4.1	1.75	2.2
Travelled Distance (m)	4.047	4.111	4.203	4.496	8.258
Distance error %	0.3	1.9	4.2	11.5	104