Nonlinear Control of a differential wheeled mobile robot in real time-Turtlebot 2

Chaima Bensaci, Youcef Zennir, Denis Pomorski

Nonlinear Control of a Differential Wheeled Mobile Robot in Real Time -Turtlebot 2

Chaima BENSACI [1], Youcef ZENNIR[2], Denis POMORSKI[3]
(1) LGCES Laboratory Skikda, Algeria
(2)Automatic Laboratory of Skikda, Algeria
(3) CRIStAL Laboratory– UMR 9189 Lille, France
[1]ch.bensaci@univ-skikda.dz; [2]y.zennir@univ-skikda.dz; [3] denis.pomorski@univ-lille1.fr

Abstract – *The paper presents a trajectory tracking controller for a non-holonomic differential wheeled mobile robot (DWMR) using the robot kinematic model. A description of the robot is presented followed by a hard robot model and a brief presentation of the ROS system to control the real robot thereafter. The nonlinear controller that provides stability according to Lyapunov function is formulated followed by a 3D simulation under Gazebo software and a set of experiments was carried out on a real Turtlebot 2 for different types of trajectories. Finally we presented the results which clearly show the effectiveness and the efficiency of our control approach even for complex paths followed by prospects for future work.*

Keywords: *Modeling, non linear system control, Real time experimentation; Trajectory tracking; Gazebo environment; differential wheeled mobile robot; Turtlebot 2 robot.*

I. INTRODUCTION

The motion control of wheeled mobile robots has been, and still is, the most attractive research works [1-7] due to their non-holonomy constraints which motivate the improvement of highly nonlinear control techniques. Among the most frequent problems in the robot motion control is trajectory tracking. Trajectory tracking is concerned with the controller conception to force a WMR to track a geometric path with an associated timing law. Many research have been conducted to study trajectory tracking control problem of WMR with several control strategies for example, adaptive control [1-17], backstepping control [12], feedback linearization [16], fuzzy logic control [13,18,19], neural network control [20], Fuzzy adaptive iterative learning control [21]. A survey on various motion control problems of WMR is done in [14, 15].

In this paper, we have chosen to work with the kinematic model. We have used different forms of trajectories and tried to control a differential wheeled mobile robot DWMR named Turtlebot 2 by delivering velocity commands as shown in the figure 1.

In this paper, the work is organized as follows: sections 1and 2 include introduction and brief presentation of the Turtlebot 2. Next, section 3 and 4 present the robotic operating system and Turtlebot 2 control via MATLAB. In section 5 the Turtlebot 2 control for tracking different trajectories using odometric estimation data is presented. Both 3 D gazebo simulation result and real time simulation result are mentioned in section 6. Finally, we conclude in section 7.

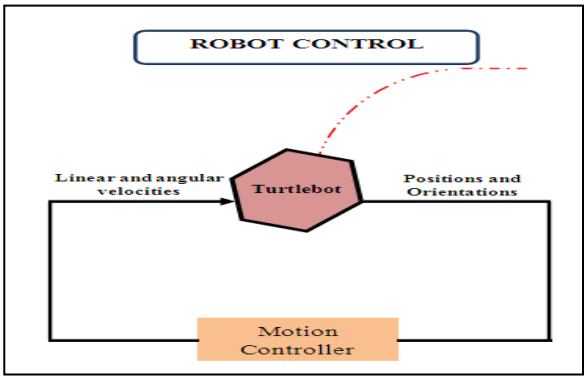

Fig.1. Controlling a robot to track a desired trajectory

II. BRIEF PRESENTATION OF THE TURTLEBOT 2

The TurtleBot 2 is one of the non-holonomic robots officially proposed by Willow Garage to develop in the operating system dedicated to robotics: ROS. It is equipped with a Kinect sensor, a Netbook, trays for the installation of these two components and a Kobuki base as shown in figure 1.

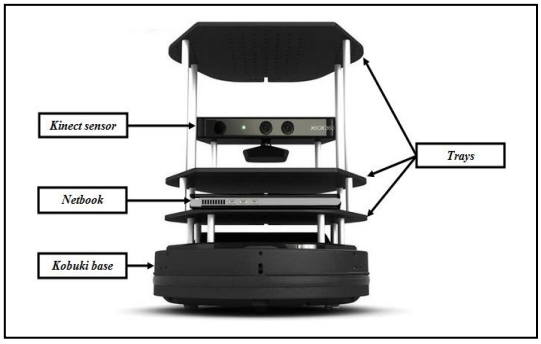

Fig.2. Controlling a robot to track a desired trajectory

The robot computer : The main task of the robot computer is to receive data and send it to a desktop computer or save it to a hard disk. The robot computer also interacts with the TurtleBot hardware.

The Kobuki base : The Kobuki is the Turtlebot 2 mobile base. It is a mobile wheeled robot with two differential wheels and two castor wheels, it also contains proximity sensors, encoders on the wheels and a gyrometer for each axis [8]. The Kobuki base can also supply power to external sensors (Kinect, ultrasonic sensors, infrared sensors, laser scanners, other cameras ...) or to actuators (motors, servomotors). [9] The following figure shows the top and bottom Kobuki's views.

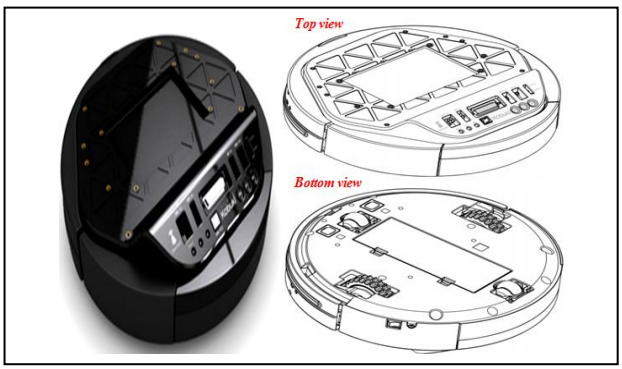

Fig.3. The kobuki base [10]

- **Kobuki base sensors**

- **Encoders**: Encoders are sensors attached to a rotating object (eg wheels or motors) to measure rotation. By measuring the rotation, the displacement, speed and acceleration of the robot can be determined. These encoders allow location by odometry. The odometry is based on the individual measurement of wheels movements to reconstruct the overall movement of the robot. Starting from a known initial position and integrating the measured displacements, it is then possible to calculate at each instant the current position of the mobile robot.

- **Gyroscope**: The gyroscope is a sensor of angular velocities on three axes x, y and z.
- **Bumpers**: There are three bumpers which are distributed between the left part, the center and the right part of the base.
- **Vacuum sensors**: Similarly, there are three sensors which are distributed between the left, the center and the right part of the base.
- **Wheel descent sensor**: There are two of these sensors one for each wheel (left, right). ***Kinect sensor :*** The figure below show the kinect sensor.

Fig.4. The kobuki base

The main Kinect components are:

- RGB (Red Green Blue) camera that stores data in three channels with a resolution of 1280x960 pixels. It allows the capture of a color image.

- Infrared transmitter / receiver (IR) and IR depth sensor. The transmitter emits beams of IR light and the depth sensor reads the IR beams reflected by the obstacles encountered. The reflected beams are converted into depth information thus measuring the distance between the obstacle and the sensor. This technology allows the capture of a depth image.

III. ROBOTIC OPERATING SYSTEM (ROS)

The ROS system is a nice interface between the robot and the man. It is a software patch implemented on Linux / Ubuntu system. The ROS software is organized into packages. A package includes many nodes (nodes). The nodes communicate with each other by passing messages. Therefore, the node must name the message it wants to publish or subscribe to the ROS core, called the master. Communication between the nodes can be either synchronous with the services, or asynchronous with the topics.

IV. TURTLEBOT 2 CONTROL VIA MATLAB

In order to control the Kobuki base of the Turtlebot 2 remotely through a computer called here "Workstation", it is necessary to create a Master-Slave communication relationship between the Workstation and the netbook which is equipped by Ubuntu /ROS.

- ***Creating a Master / Slave Relationship***

First, to create this relationship between the two computers, they must be defined on the same network. The Workstation must be identified as a Master from MATLAB using the following commands :

>> setenv ('ROS_IP', 'The IP address of the MASTER')
>>rosinit

The "rosinit" command creates a global ROS node that communicates with the ROS node of the netbook. To stop the process, we use the command "rosshutdown" .The configuration of the robot computer is done from its terminal (Linux operating system) by the following steps:

We have to open the bashrc file in the console with the command nano.bashrc. Then insert the following commands:
>> report ROS_MASTER = The IP address of the master: the port number (11311)
>> report ROS_HOSTNAME = The IP address of the robot's computer

After saving the file bashrc, we source it by the command: source.bashrc. Then we launch the Turtlebot with thecommand:
>> roslaunch turtlebot_bringup minimal.launch

- ***Velocities publication***

In order to control the Turtlebot, we must register as a publisher on the velocity command topic from MATLAB and create a message in the format used by this topic. Finally, we give values to velocities and send the message.

Adjustable velocities on the Turtlebot are linear X velocity and angular Z velocity. They allow the robot to move forward and turn. The other parameters of the linear and angular velocities are zero.

- ***Sensors data acquisition***

To acquire sensors data (odometer, gyroscope...) through Matlab, it is necessary to act in a similar way to the commands of the terminal. First, we subscribe to the topic as subscriber and we create a message variable to retrieve the messages.

V. CONTROL OF TURTLEBOT 2 FOR TRACKING DIFFERENT TRAJECTORIES- ODOMETRIC ESTIMATION DATA

V.1. The kinematic Model of the Robot

The Turtlebot 2 is moved by the following model: We consider that (X, Y) is the global coordinate system and (XR, YR) is the local coordinate system of the robot. The position of the robot P (x, y, θ) is represented in Cartesian coordinates in the global coordinate system. The relationship between the robot frame and the global frame is given by the basic transformation matrix:

$$R(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (1)$$

The figure below show the relationship between the two frames.

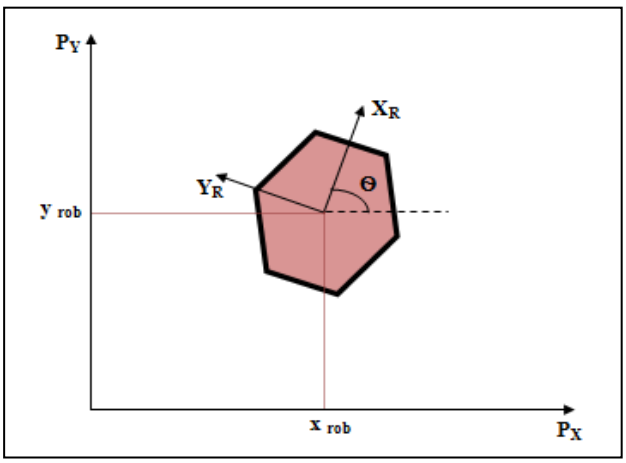

Fig.5. Turtlebot 2 representation in a Cartesian coordinate system

The wheels are motorized independently. When both wheels turn with the same speed in the direct action the robot moves forward otherwise it moves backwards. Turning right is done by actuating the left wheel at a higher speed than that of the right wheel and vice versa to turn left. It also can rotate on the spot by actuating a wheel forward and the second wheel in the opposite direction with the same speed. The third wheel is a free wheel preserve the robot stability.

The Turtlebot 2 wheels are non-holonomic, they represent non holonomic constraints:

$$\dot{x}\ \sin\theta\ - \dot{y}\ cos\theta = 0 \quad (2)$$

The relationship between the speeds of the robot (v, w) and the speeds of the wheels (wr, wl) is expressed by this two equations:

$$v = v_x\ =\ v_1 +\ v_2 = r(\tfrac{w_r + w_l}{2}) \quad (3)$$

$$w = w_1\ +\ w_2 = r(\tfrac{w_r - w_l}{d}) \quad (4)$$

Where r is the wheel radius and d is the distance between wheels.

V.2. Kinematic Equations

Suppose that the robot is in an arbitrary position P (xc, yc, θ) and the distance between its current position and the desired position R (xd, yd, β) ; defined with reference to the global frame; higher than 0.

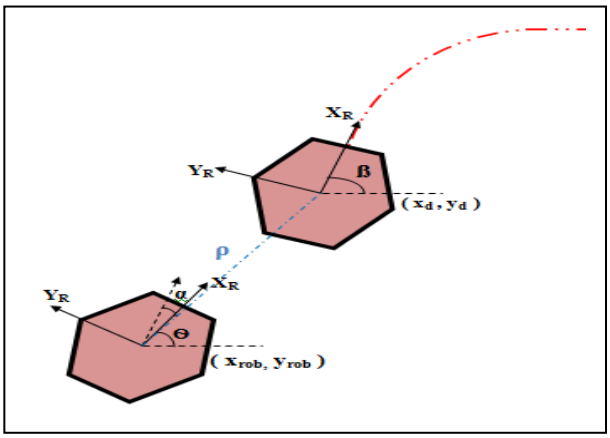

Fig.6. Turtlebot 2 position and orientation error vectors

The robot Cartesian is given by:

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = w \end{cases} \quad (5)$$

Where : x, y and θ are measured relative to the global frame.

The robot position can also be represented by the polar coordinates with a distance error ρ>0:

$$\begin{cases} \dot{\rho} = -v\cos(\beta - \theta) = -v\cos\alpha \\ \dot{\beta} = v\ \frac{\sin\alpha}{\rho} \\ \dot{\theta} = w \end{cases} \quad (6)$$

where $\alpha = \beta - \theta$ is the angle measured between the axis X_R of the reference relative to the robot and the axis ρ expressed by the following equation:

$$\rho = \sqrt{(xd - xc)^2 + (yd - yc)^2} \quad (7)$$

The kinematic equations obtained on the basis of the polar coordinates are:

$$\begin{cases} \dot{\rho} = -v\cos\alpha \\ \dot{\alpha} = -w + v\ \frac{\sin\alpha}{\rho} \\ \dot{\beta} = v\ \frac{\sin\alpha}{\rho} \end{cases} \quad (8)$$

This set of equations is only valid for $\rho \neq 0$.

V.3. Control Law and Stability

The control algorithm must be designed to move the robot from its current configuration to its desired configuration, so we are going to send linear and angular velocity commands $u = \begin{pmatrix} v \\ w \end{pmatrix}$ until the robot reaches the desired position.

The proposed control law is state-dependent ie. $\begin{pmatrix} v \\ w \end{pmatrix} = f(\rho, \alpha, \beta)$, which ensured that the state is moving toward (0, 0, β) without reaching ρ = 0 in a finite time interval. The following figure represents the control structure.

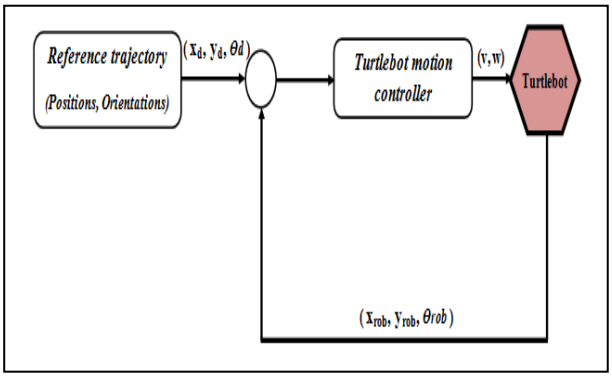

Fig.7. The proposed control structure

One of the most commonly used methods for studying asymptotic behavior is based on Lyapunov stability theory. Consider a simple positive definite quadratic form of the Lyapunov function:

$$V = V_1 + V_2 = \frac{1}{2}\rho^2 + \frac{1}{2}\alpha^2 \quad (9)$$

ρ *et* α are respectively the distance and orientation errors.

Its derivative relative to time is given by:

$$\dot{V} = \dot{V_1} + \dot{V_2} = \dot{\rho}\rho - \dot{\alpha}\alpha \quad (10)$$

Using the kinematic equations:

$$\dot{V} = (-v\cos\alpha) + \alpha\,(-w + v\,\frac{\sin\alpha}{\rho}) \quad (11)$$

The first term can be non-positive, putting the linear velocity in the form:

$$\begin{cases} v = K_\rho\,\rho\cos\alpha \;\; avec \;\; K_\rho > 0 \\ \dot{V_1} = \rho\,(-K_\rho\,\rho\cos^2\alpha) \\ \dot{V_1} = -K_\rho\,\rho^2\cos^2\alpha \le 0 \end{cases} \quad (12)$$

This means that the term $\dot{V_1}$ is always non-increasing over time, therefore it converges asymptotically to a non-negative finite limit. Likewise, the second term can be non-positive thus the angular velocity is put in the form:

$$w = K_\rho\sin\alpha\cos\alpha + K\alpha \;\; avec \;\; K\alpha > 0 \quad (13)$$

$$\dot{V_2} = \alpha\,(-K_\rho\sin\alpha\cos\alpha - K\alpha\,\alpha - \frac{K_\rho\,\rho\,\sin\alpha\cos\alpha}{\rho}) \quad (14)$$

$$\dot{V_2} = -K\alpha\,\alpha^2 \le 0 \quad (15)$$

Finally, the expression of the time derivative of the Lyapunov function *V* becomes:

$$\dot{V} = \dot{V_1} + \dot{V_2} = -K_\rho\,\rho^2\cos^2\alpha - K\alpha\,\alpha^2 \le 0 \quad (16)$$

The result is in semi-negative form. By applying the Barbalat lemma, it follows that *V* necessarily converges to zero on time. Which implies that the state vector convergence from (ρ, α, β) to (0, 0, β).

So we conclude that the expressions of the following linear and angular velocities make the movement of the robot smooth and stable [8].

$$\begin{cases} v = K_\rho\,\rho\cos\alpha \\ w = K_\rho\sin\alpha\cos\alpha + K\alpha\,\alpha \end{cases} \quad (17)$$

V.4. Reference trajectories and sensor data acquisition

The reference trajectory can give the destination according to the current position and the trajectory. The controller is tested on different reference trajectories: circle, infinite, butterfly, line and other paths. We used the data recovered by the odometer to estimate the position along the x and z axes and the orientation of the robot at every moment.

VI. SIMULATION AND REAL TIME RESULTS

VI.1. Gazebo 3 D Simulation Result

Gazebo is a robotic three-dimensional simulator for indoor and outdoor environments. it is capable of simulating a population of robots, sensors and objects.

It generates both realistic sensor feedback and physically plausible interactions between objects (it includes an accurate simulation of rigid body physics).

By realistically simulating robots and environments code designed to operate a physical robot can be executed on an artificial version. Numerous researchers have also used Gazebo to develop and run experiments solely in a simulated environment.

Gazebo simulation can be regarded as a experimental copy of real robot in virtual world [11].

In our work we use the simulated turtlebot2 of gazebo world, the figure below show the turtlebot 2 simulated robot under gazebo world.

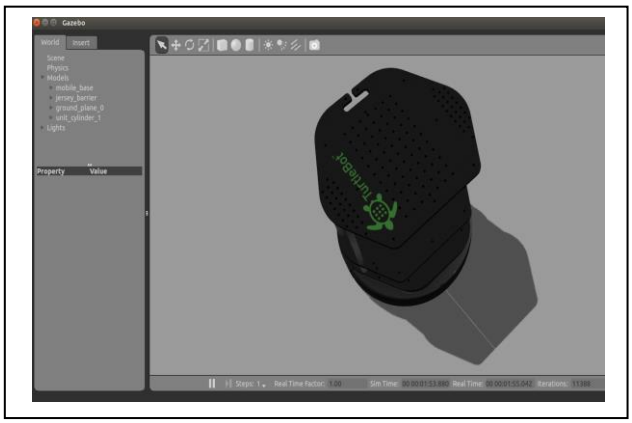

Fig.8. The simulated Turtlebot 2 in gazebo world

At the end of the 3 D gazebo simulation for different trajectories, the following results were obtained (figures 9,10,11,12, 13):

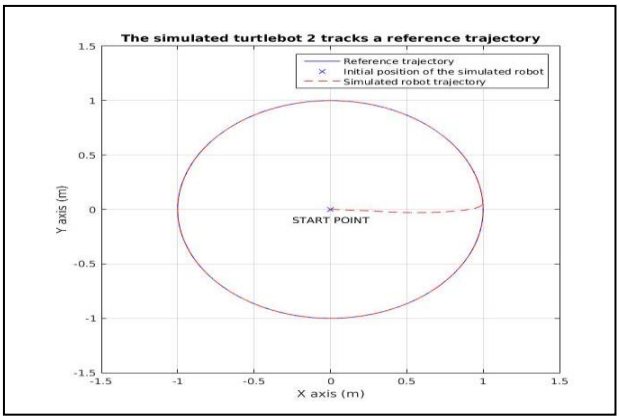

Fig.9. Simulated Turtlebot 2 tracks a circular reference trajectory

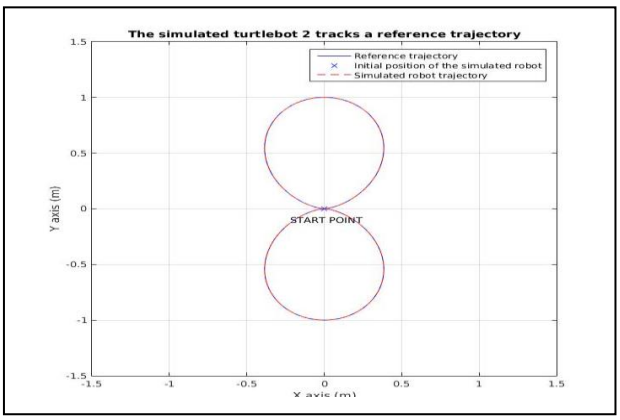

Fig.10. Infinite trajectory following by the simulated Turtlebot2

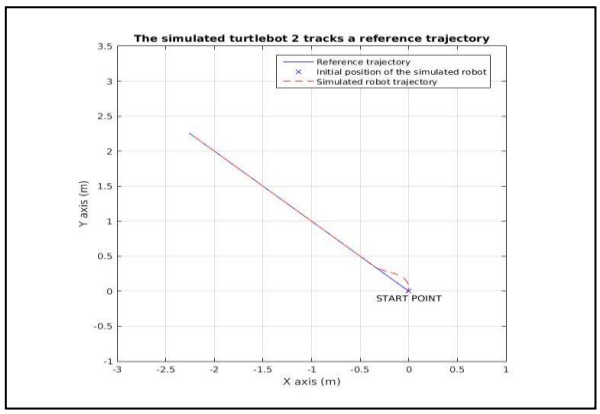

Fig.11. Inclined straight trajectory tracked by the simulated Turtlebot 2

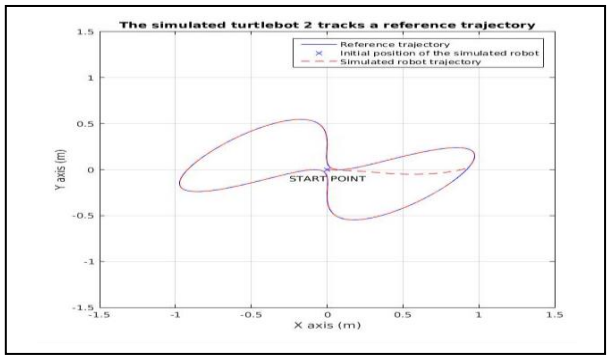

Fig.12. The simulated Turtlebot 2 track a desired trajectory

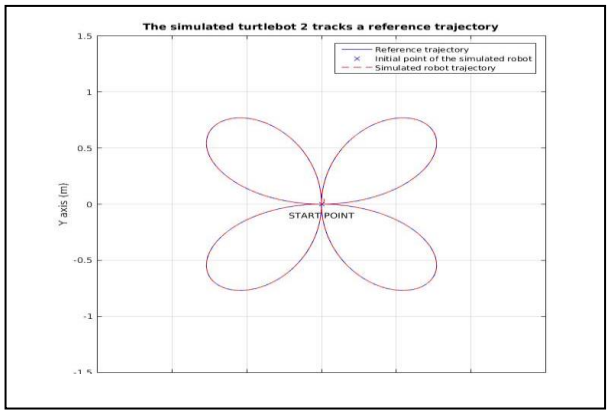

Fig.13. The simulated Turtlebot 2 track a desired trajectory

The follows figures 14, 15,16,17,18 illustrated the obtained results with real time simulation in Turtlebot2 robot.

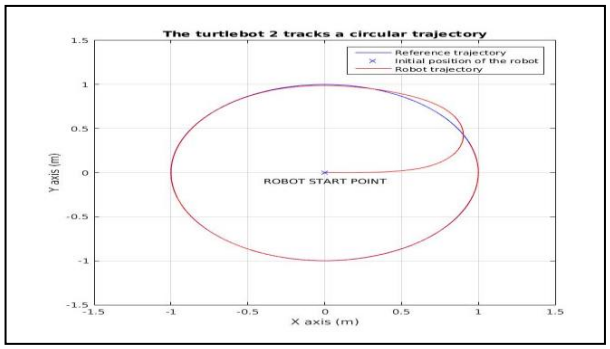

Fig.14. Circular trajectory tracked by Turtlebot 2

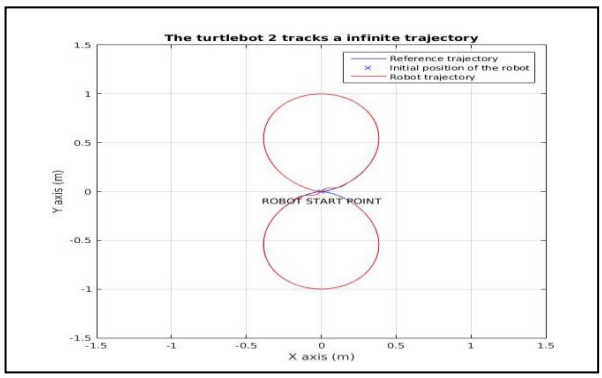

Fig.15. Infinie trajectory tracked by Turtlebot 2

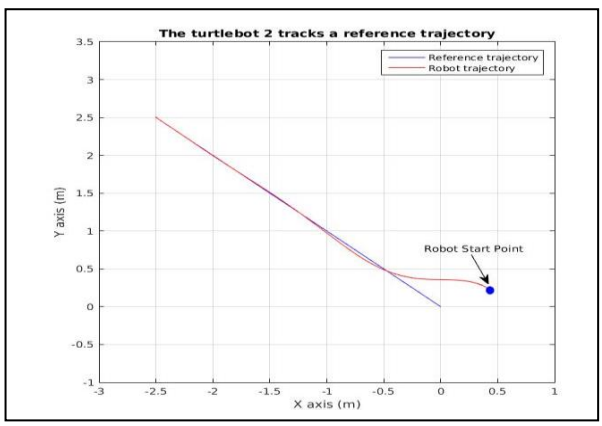

Fig.16. Straight line trajectory tracked by Turtlebot 2

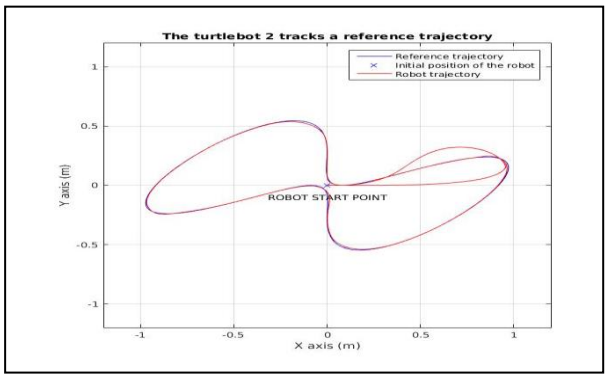

Fig.17. Reference trajectory tracked by Turtlebot 2

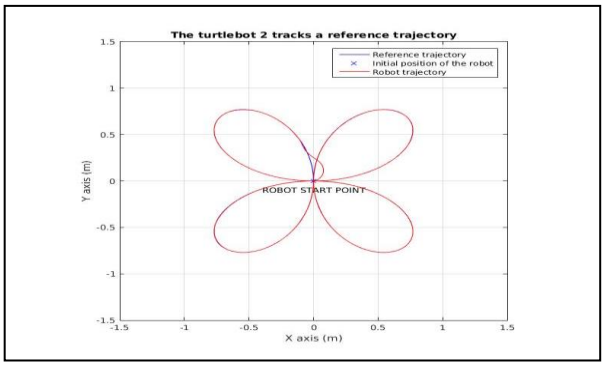

Fig.18. Reference trajectory tracked by Turtlebot 2

VI.2. DISCUSSION OF OBTAINED RESULT

The Turtlebot 2 tracks perfectly the reference trajectories either in simulation or in real time with sampling time = 0.5 ms for the data transfer, a maximum linear velocity V max = 0.4 m / s and a maximum angular velocity Wmax = 0.4 m / s.

The trajectory of the real robot is close to the reference trajectory with a position error of 1 mm which verifies the control law used as illustrated in figure 19.

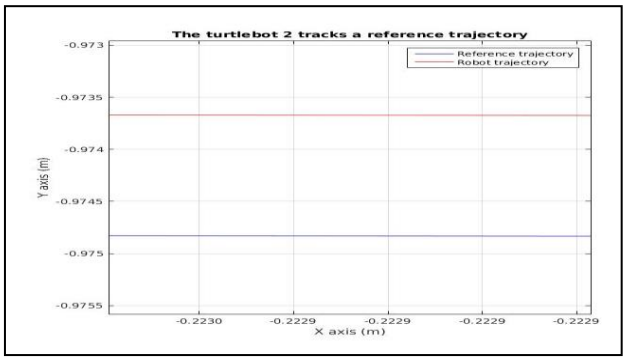

Fig.19. Reference trajectory and real robot trajectory.

In simulation, we obtained trajectories much closer to the reference trajectory with a position error of 0.01 mm as shown in the figure 20.

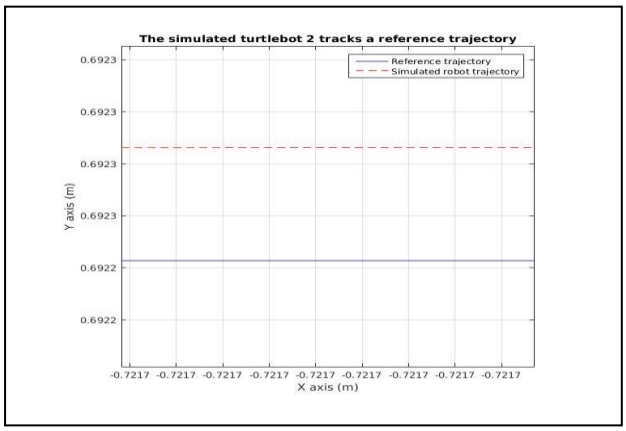

Fig.20. Reference trajectory and simulated robot trajectory

With the different obtained results in the different simulation we can observe clearly the efficiency of our control approach either in simulation or in real time on the robot. The stabilization time and position error is very accepted in the different trajectory. But the obtained results with simulation in gazebo are more precise than in real time because of the faster data transmission frequency of simulator.

VII. CONCLUSION

In this paper, the trajectory tracking problem is solved, we are used a differential wheeled mobile robot DWMR (Turtlebot 2) to track different reference trajectories: circular, infinite, line…with its kinematic model. Simulations are performed with different trajectories to evaluate the performance of the proposed controller for both simulated and real Turtlebot 2. We are improved the performance of our controller in 3D gazebo simulation and in real experiment. We are described the control strategy used to control robot motion. The obtained results show the efficiency and the effectiveness of the motion control strategy. The simulation results are more precise than in real time because of the faster data transmission frequency of simulator. In the future work we will approve and generalize this study with two or more robots with the use of artificial intelligence techniques like genetic algorithm, PSO, Q-learning algorithm, fuzzy logic control, Q-fuzzy algorithm….

REFERENCES

[1] K Shojaei, AM Shahri, A Tarakameh, B Tabibian , Adaptive trajectory tracking control of a differential drive wheeled mobile robot, ROBOTICA 2011, pages:1-12, Cambridge university.

[2] Zhixi Shen, Yaping Ma and Yongduan Song, Robust Adaptive Fault tolerant Control of Mobile Robots with Varying Center of Mass, IEEE Transactions on industrial electronics, 2017.

[3] Xinwu liang, hesheng wang, Formation Control of Nonholonomic Mobile Robots Without Position and Velocity Measurements IEEE TRANSACTIONS ON ROBOTICS, VOL. 34, NO. 2, APRIL 2018.

[4] P. Morin and C. Samson, Trajectory tracking for non-holonomic vehicles:overview and case study, Fourth International Workshop on Robot Motion and Control, lune 17-20,2004.

[5] C. Samson, "Control of chained systems application to path following and time-varying point-stabilization of mobile robots," *IEEE Trans. Autom. Control* (1), 64–77 (1997).

[6] C. C. de Wit and H. Khennouf, "Quasi-Continuous Stabilizing Controllers for Nonholonomic Systems: Design and Robustness Considerations," *Proceedings of the 3rd European Control Conference*, Rome, Italy (1995) pp. 2630–2635.

[7] M. Oya and R. Chun-Yi Su Katoh, "Robust adaptive motion/force tracking control of uncertain nonholonomic mechanical systems," *IEEE Trans. Robot. Autom.* **19**(1), 175–181 (2003).

[8] F.WEHBI, "Architecture logicielle et matérielle pour fusion tolérantes au fautes pour systèmes multi-robots sous ROS". Rapport, Lille and Libanon University, 2015, pp.64.

[9] Bilal DAASS, Cartographie automatisée d'une structure à partir d'un radar bi-statique micro-ondes embarqué sur robots mobiles collaboratifs, projet d'ingéniorat, Informatique industrielle, Université Libanaise, 74 pages, 2017.

[10] Dabit Industries, kobuki Documentation Release 2.0, 43 pages, Oct 01, 2017.

[11] Wei Qian, Zeyang Xia, Jing Xiong, Yangzhou Gan, Yangchao Guo, Shaokui Weng, Hao Deng, Ying Hu, Jianwei Zhang, Manipulation Task Simulation using ROS and Gazebo, Proceedings of the 2014 IEEE International Conference on Robotics and Biomimetics December 5-10, 2014, Bali, Indonesia.

[12] U. Kumar and N. Sukavanam, "Backstepping based trajectory tracking control of a four wheeled mobile robot," International Journal of Advanced Robotic Systems, vol. 5, pp. 403-410, Dec 2008.

[13] S.G. Tzafestas, K.M. Deliparaschos, G.P. Moustris, Fuzzy logic path tracking control for autonomous non-holonomic mobile robots: Design of System on a Chip, Robotics and Autonomous Systems 58 (2010) 1017-1027.

[14] I. Kolmanovsky and H. McClamroch, "Developments in nonholonomic control problems," *IEEE Control Syst. Mag.*20–36 (Dec. 1995).

[15] Alessandro De Luca, Giuseppe Oriolo, Marilena Vendittelli, Control of Wheeled Mobile Robots: An Experimental Overview,pp :1-46.

[16] G. Oriolo, A. De Luca, and M. Vendittelli, "WMR control via dynamic feedback linearization: Design, implementation, and experimental validation," IEEE Transactions on Control Systems Technology, vol. 10, pp. 835-852, Nov 2002.

[17] R. Barzamini, A. R. Yazdizadeh, and A. H. Rahmani, "A New Adaptive Tracking Control For Wheeled Mobile Robot," in 2006 IEEE Conference on Robotics, Automation and Mechatronics, 2006, pp. 1-6.

[18] J. Keighobadi and M. B. Menhaj, "From Nonlinear to Fuzzy Approaches in Trajectory Tracking Control of Wheeled Mobile obots," Asian Journal of Control, vol. 14, pp. 960–973, 2012.

[19] H. Yu, G.-Y. Tang, H. Su, C.-P. Tian, and J. Zhang, "Trajectory tracking control of wheeled mobile robots via fuzzy approach," in Control Conference (CCC), 2014 33rd Chinese, 2014, pp. 8444-8449.

[20] J. Ye, "Tracking control of a non-holonomic wheeled mobile robot using improved compound cosine function neural networks," International Journal of Control, vol. 88, pp. 364-373, Feb 1 2015.

[21] Xiaochun Lu, Juntao Fei1, Velocity Tracking Control of Wheeled Mobile Robots by Fuzzy daptive Iterative Learning Control, 28th Chinese Control and Decision Conference (CCDC), IEEE , pp: 4242-4247, 2016.