IJCTA, 9(2-A), 2016, pp. 405-422
© International Science Press

Design and Control with Improved Predictive Algorithm for Obstacles Detection for Two Wheeled Mobile Robot Navigation

Sabri M. Ben Mansour*, V. Sundarapandian and Saber M. Naceur***

ABSTRACT

This paper addresses the path following control problem of a non holonomic Two-Wheeled Inverted Pendulum Mobile Robot. We propose a control architecture based on two control layers. A speed innerloop control scheme is first designed based on state feedback technique to ensure stability of the inverted structure of the robot. A second outerloop control scheme is proposed to help the TWIPMR navigate along a desired path formed by a set of way points. It is designed inspiring the MPC technique. The elements of the predictive control, which are the cost function, controls and constraints, must be defined and specified: the use of different trajectories group in the control can adapt the behavior of the robot to different displacement phases.

Keywords: Mobile robot; navigation; stability; Predictive control

1. INTRODUCTION

The problem of non-holonomic control systems has interested many researchers lately. The most studied example, with significant, experimental and very varied results, is the two wheeled mobile robot which the kinematic model is identical to a unicycle[1-3]. The differentially driven robots that are frequently presented in practical applications also have similar kinematic model. Likewise many researchers coped with the more difficult problem of stabilizing dynamic models for different types of mobile robots[4-7]. Robot control constraints are usually tied to its kinematic model, which is very important from the implementation point of view as the material used for the control of the wheel's speed is generally a simple micro-controller.

In the literature, the robot control has-been approached by speed stability or by a tracking control problem. There are other works that studied both approaches simultaneously. We think that the second approach is more interesting, since the kinematic constraints and other control aims (Obstacle avoidance, proceeding time and minimum energy consumption) are implicitly encompassed in the path-planning process[8-9].

The first path planning method consists in the robot environment decompose in cells [10] (in a set of adjacent connected regions). Then simply find an algorithm working on the quantification of the environment. There, different techniques exist, such as the Voronoi partitions [11] or the visibility graphs [12]. In these cases, the discrete environment is represented in a graph and the trajectory is generated by finding a path in the graph, this problem can be easily processed by computer [13].

Other planning algorithms work in a discrete environment. Among the best known A * [14] which is planning an optimal path knowing the environment or D * [15] which is planning a path dynamically.

* LTSIRS, Remote Sensing and Spatially Referenced Information Systems Laboratory, National Engineering School of Tunis, Tunisia, *Email: sabri.benmansour@gmail.com*

** Research Centre, Vel Tech University, Avadi, Chennai-600062, Tamil Nadu, INDIA

However, these algorithms have large defects. Including the fact that the kinematic constraints of the robots are not considered which allows the planner to find a path that will not be executed by the robot. Moreover, the fact of discretizing the search space limits the possible trajectories. The Michael Defoort algorithm [16] generates pieces of polynomial splines that meet the kinematic constraints of the robot. The algorithm can be simplified by forcing it to generate not splines, but simply polynomials. The trajectory of the robot is then assimilated with the juxtaposition of several polynomials. A fuzzy model of a predictive control was proposed by [17]. Interesting work from [18] was about the design of an optimal trajectory wich is made based on Bernstein-bézier curve and optimization. Some recent control methods are discussed in [28-30].

We are looking to developa control scheme, as the paths optimization problem, and adapt them to our navigation process. The aim was to get an under constraints optimization algorithm which can be implemented in a real-time application. The navigation method that we propose follows several steps. In order to enable the robot to join the path provided by the modeling environment, a catch curve is generated.

The robot is controlled on a path by considering two deviation parameters: the distance to the trajectory and the difference between the robot angle and the straight line tangent to the curve's path. These two regulators in cascade made it possible to follow a path divided in multiple straight lines to embody the desired trajectory. Obstacles, when identified, are represented by the intersection of two straight lines. The anticipation movement from a line two another is made through the predictive control taking the angle of these two lines as manipulated variable.

Based on the knowledge of the robot's model and its constraints, feasible trajectories can be generated on a given time horizon. A constraint optimization algorithm will then determine the closest feasible path by the robot to the reference trajectory under constraints that we have developed. Finally, the generated control law is provided to the robot actuators, and applied according to a chosen sampling time before reconsidering all the navigation process reinitiating.

The formalization of the navigation problem in this form is a significant contribution over the existing methods, for example by direct model. Simulations and implementations have validated our method by showing its effectiveness to generate achievable trajectories by the robot.

2. PROBLEM STATEMENT

Considering a differentially driven, two-wheeled mobile robot like the one showed in Fig. 1, where (x, y) is the wheel-axis-centre position and θ is the motion orientation. The kinematic motion equations of such a mobile robot are similar to a unicycle model. This kind of robot has a nonholonomicconstraint of the form

$$\dot{y} \cdot \cos\theta(t) - \dot{x} \cdot \sin\theta(t) = 0 \tag{1}$$

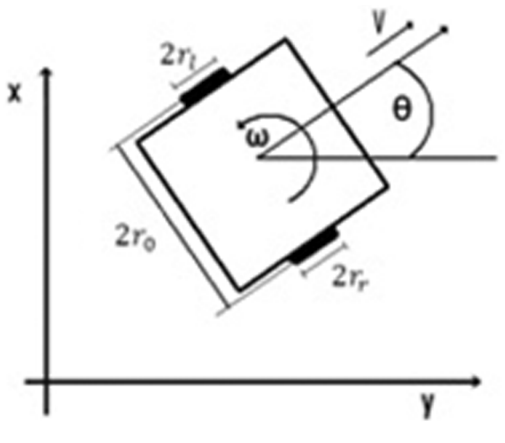


Figure 1: Mobile inverted pendulum

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \frac{r_r}{2}\cos(\theta)\frac{r_l}{2}\cos(\theta) \\ \frac{r_r}{2}\sin(\theta)\frac{r_l}{2}\sin(\theta) \\ \frac{r_r}{2r_0}\frac{r_r}{2r_0} \end{pmatrix} \cdot \begin{pmatrix} \dot{\theta}_r \\ \dot{\theta}_l \end{pmatrix} \tag{2}$$

Where $\dot{\theta}_r(t)$ and $\dot{\theta}_l(t)$ are theangular velocity of the right wheeland theangular velocity of the left wheel. By discretizing the kinematic model and using the usual approximations, the odometric model becomes

$$\theta(k) = \frac{r(\theta_r(k) - \theta_l(k))}{2r_0} \tag{3}$$

$$x(k) = x(k-1) + \frac{r(\Delta\theta_r + \Delta\theta_l)}{2} \cdot \cos\left(\frac{e(k)}{2}\right) \tag{4}$$

$$y(k) = y(k-1) + \frac{r(\Delta\theta_r + \Delta\theta_l)}{2} \cdot \sin\left(\frac{e(k)}{2}\right) \tag{5}$$

where

$$e(k) = \theta(k) - \theta(k-1) \tag{6}$$

$$\Delta\theta_r = \theta_r(k) - \theta_r(k-1) \tag{7}$$

$$\Delta\theta_1 = \theta_1(k) - \theta_1(k-1) \tag{8}$$

where

θ_r : *The angular position of the right wheel*

θ_l : *The angular position of the left wheel*

The robot has to move on the field with simple routes and in an effective way.Navigation problem is to determine a feasible path by the robot, which does not make it collide with obstacles, and to enable it to better follow the path provided by the planner. It means making a selection in a path that the robot is able to perform, and therefore determining a criterion to make that choice.

Kinematic constraints such as saturations of actuators or stops joint must be stated clearly. This type of constraint will play on input functions of our robot, on its u command. The set of trajectories that satisfy these constraints is called the set of eligible trajectories.The typical path to follow is a series of straight lines. The robot has to anticipate the next turnwith a radius of curvature, to be rotatable about an obstacle.

3. THE KINEMATIC MODEL AND THE CONTROL DESIGN

It is possible to model the system described above by the Newton-Euler method. The necessary equations are the motor mechanical equation and the robot motion equations.

3.1. Two wheeled mobile robot modeling

3.1.1. Modeling assumptions

The motor correctly filter the PWM. The inertia of the motor itself is neglected compared to the inertia of the wheel, the motor inductance is neglected and both motors have identical parameters.

3.1.2. Mechanical equation of the motor

Consider the general characteristics of the engine we find

$$U_i(t) = R_i \cdot i_i(t) \tag{9}$$

$$\ddot{\theta}_i(t) \cdot I_i = \sum_{i=1}^{n} M_i^j(t) \tag{10}$$

$$\sum_{i=1}^{n} M_i^j(t) = M_i^1(t) + M_i^2(t) + M_i^3(t) \tag{11}$$

$$M_i^1(t) = k_{i\cdot} n_{i\cdot} i_i(t) \tag{12}$$

$$M_i^2(t) = -F_i(t) \cdot r_i \tag{13}$$

$$M_i^3(t) = -f_i \cdot \dot{\theta}_i(t) \tag{14}$$

3.1.3. Motion equations of the robot

The moment of inertia is on the axis center of the wheels and gives

$$I_0 \cdot \ddot{\theta}_0(t) = r_0 \cdot \left(F_r(t) - F_l(t)\right) \tag{15}$$

$$m_0 \cdot \ddot{x}(t) = \sum_{n=0}^{N} F = F_r(t) - F_l(t) \tag{16}$$

$$\ddot{\theta}_0(t) = \frac{\ddot{\theta}_r(t) \cdot r_r - \ddot{\theta}_l(t) \cdot r_l}{2.r_0} \tag{17}$$

$$\ddot{x}_0(t) = \frac{\ddot{\theta}_r(t) \cdot r_r + \ddot{\theta}_l(t) \cdot r_l}{2} \tag{18}$$

The equations presented above show that this is a 4th order system

$$\begin{aligned} \dot{X} &= A \cdot X + B \cdot U \\ Y &= C \cdot X \end{aligned} \tag{19}$$

Then we get the explicit equations

$$\begin{pmatrix} \dot{\theta}_r \\ \ddot{\theta}_r \\ \dot{\theta}_l \\ \ddot{\theta}_l \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & a_r^r & 0 & a_l^r \\ 0 & 0 & 0 & 1 \\ 0 & a_r^l & 0 & a_l^l \end{pmatrix} \cdot \begin{pmatrix} \theta_r \\ \dot{\theta}_r \\ \theta_l \\ \dot{\theta}_l \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ b_r^r & b_l^r \\ 0 & 0 \\ b_r^l & b_l^l \end{pmatrix} \cdot \begin{pmatrix} u_r \\ u_l \end{pmatrix} \tag{20}$$

$$\begin{pmatrix} \theta_r \\ \theta_l \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \theta_r \\ \dot{\theta}_r \\ \theta_l \\ \dot{\theta}_l \end{pmatrix} \tag{21}$$

where

θ_r	:	*The angular position of the right wheel*
θ_r	:	*The angular position of the left wheel*
$\ddot{\theta}_r$	:	*The angular velocity of the right wheel*
$\ddot{\theta}_l$	:	*The angular velocity of the left wheel*

The details for Eqs (20) and (21) are not shown here and can be found in appendix 2.

The state model can then be discretized to get the form below with system sampling period chosen to 10 ms.

$$\begin{aligned} X(k+1) &= \Phi \cdot X(k) + \Gamma \cdot U(k) \\ Y(k) &= C \cdot X(k) \end{aligned} \tag{22}$$

Where Φ, Γ, C and D are matrices giving by Matlab calculation.

3.2. Control design

In the literature, the control law for the tracking path is basedon the assumption that the robot speed limits are directly transmitted[19]-[20]. This hypothesis corresponds to a control in perfect speed.

This separation between low-level (robot speed control) and High-level (tracking path) in Fig. 2 appears to be a good solution for many reasons, appropriate segregation of duties for readability, best potential for reusability and debugging easier when implementing.

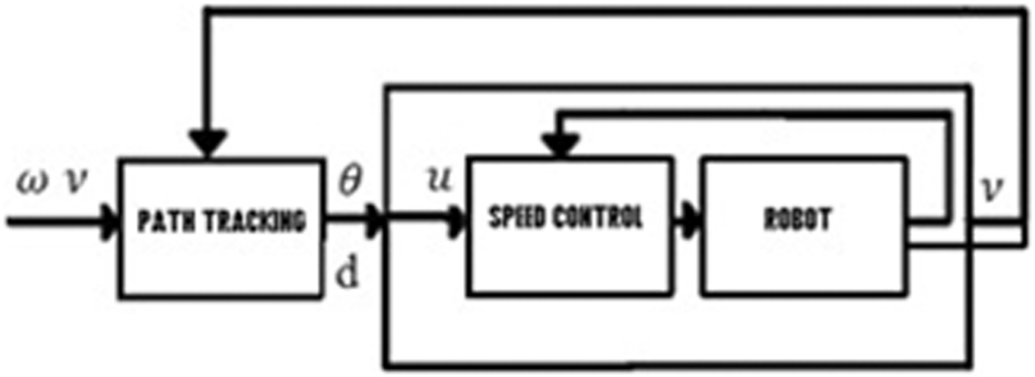


Figure 2:The two controllers in cascade

4. THE SPEED AND TRAJECTORY CONTROL

4.1. Control speed

Based on the discrete state model, it is possible to synthesize a controller state. The choice fell on the optimal control [21] that responds to the needs perfectly as shown in bloc diagram Fig. 3.

The cost function to be minimized is:

$$J = \frac{1}{2}\sum_{k=0}^{N}\left[x^T(k)\cdot Q_1 x(k) + u^T(k)\cdot Q_2 u(k)\right] \tag{23}$$

We must therefore choose the matrices Q1 and Q2. They are selected positive diagonals. The weight to be given between the elements (motors, gears) right and left are logically chosen identical. The weight of the position is zero. The weight of the speeds is much higher than the voltage of engines.

$$Q_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 100'000 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100'000 \end{pmatrix} \tag{24}$$

$$Q_2 = \begin{pmatrix} 10 & 0 \\ 0 & 10 \end{pmatrix} \tag{25}$$

A design linear-quadratic state-feedback regulator was calculated by Matlab control to give the optimum gain. Thus, the voltage applied to closed loop is

$$U(k) = K_G \cdot X(k) \tag{26}$$

Where K_G containsthe gain obtained by Matlab calculation

4.2. Path Tracking

From the information provided by the planer, a remedial curve, connecting the current position of the robot and the desired position can be determined. This curve is a purely geometric, which is not related to time,

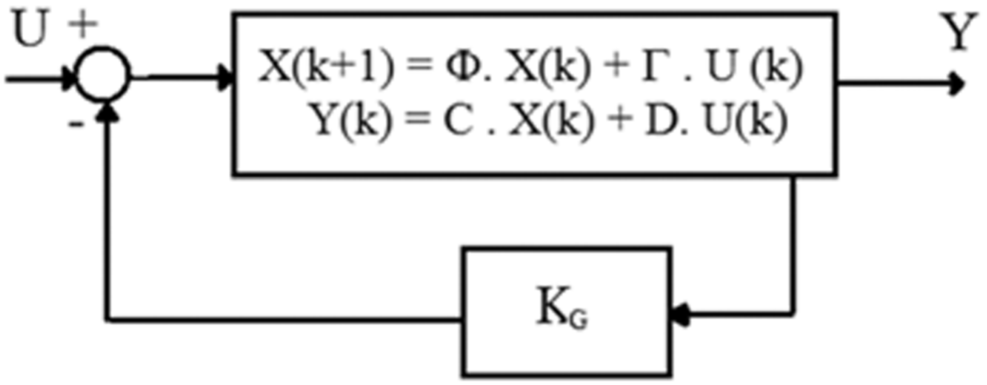


Figure 3:Block diagram of the control speed

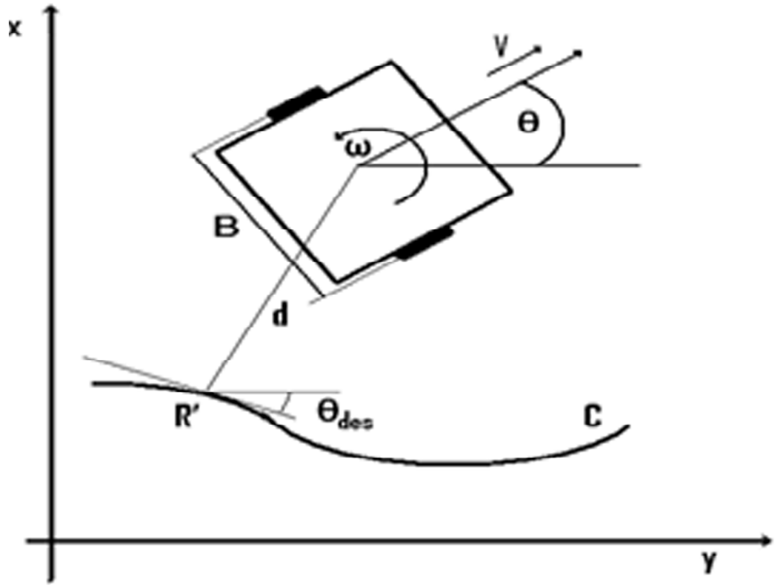


Figure 4: Path Followed by robot

but that can take into account certain geometric constraints robot (size, turning radius). Angular velocity ω = θ of the robot is used as a control variable as shown in fig. 4.

The purpose of the proposed control law is to minimizeboth the error in distance and orientation

$$\theta e = \theta - \theta des \tag{27}$$

It is given in its simplest form by

$$\omega = \upsilon \cdot \left(p\left(sR' \right) - k_1 \cdot d - k_2 \cdot \theta_e \right) \tag{28}$$

Where sR' is the curvilinear abscissa of the point R0 and k_1 and k_2 are two positive constants.

Constant values for limiting the oscillations canbe obtained in the case of a straight path

$$\begin{aligned} k_1 &= \xi^2 \\ k_2 &= 2\varsigma\xi \end{aligned} \tag{29}$$

4.3. The proposed approach of the Predictive Control

4.3.1. The cost function

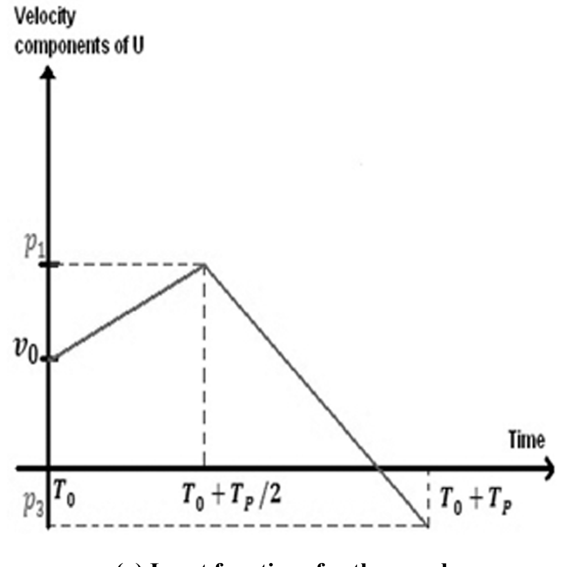
This cost function calculates the square error between the reference trajectory and the robot path, by weighting differently the various components of the state of the robot and also by weighting differently the terminal error and the tracking error of the course [23]-[24].

$$J(k) = \sum_{n=0}^{N} \left[\hat{z}(k+n) - r(k+n) \right]^T Q \left[\hat{z}(k+n) - r(k+n) \right] + \lambda \cdot \omega^2 (k+n) \tag{30}$$

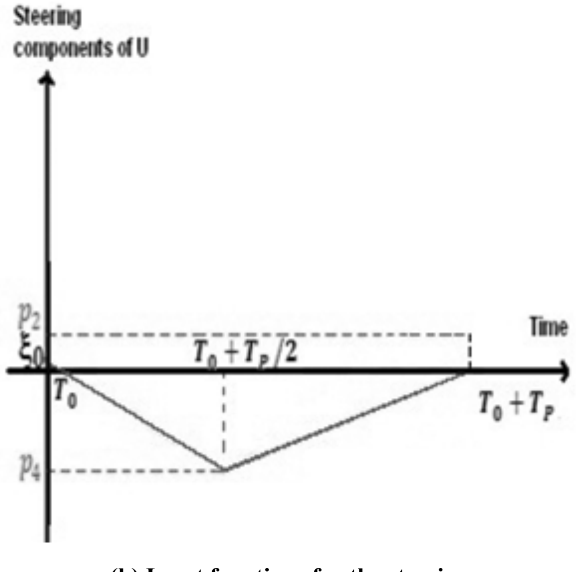
4.3.2. The control functions and the generated paths

To have good possibilities to avoid obstacles, we used linear type control functions $u(v, \xi)$ in Figs. 5 (a)-(b).

The parameters $p1$ and $p3$ denote the speed and turning respectively to achieve in $T_0 + \frac{T_P}{2}$ and $p2$ $p4$ the parameters of the speed and the turning to reach by $T_0 + T_p$. T_p is the horizon of predictions.



(a) Input functions for the speed



(b) Input functions for the steering

Figure 5: Input functions

4.3.3. Constraints

These parameters must meet a number of constraints with respect to maximum and minimum speed (v_{min}, v_{max}) and steering (ξ_{min}, ξ_{max})

$$\begin{aligned} v_{\min} &\le p_1 \le v_{\max} \\ v_{\min} &\le p_3 \le v_{\max} \\ \xi_{\min} &\le p_2 \le \xi_{\max} \\ \xi_{\min} &\le p_4 \le \xi_{\max} \end{aligned} \tag{31}$$

They must also respect the constraints on maximum acceleration ($accel_{max}$), deceleration ($decel_{max}$) and steering (ξ_{max})

$$\begin{aligned} v_0 - decel_{\max} \times \frac{T_p}{2} &\le p_1 \le v_0 + accel_{\max} \times \frac{T_p}{2} \\ p_1 - decel_{\max} \times \frac{T_p}{2} &\le p_3 \le p_1 + accel_{\max} \times \frac{T_p}{2} \\ \xi_0 - \xi_{\max} \times \frac{T_p}{2} &\le p_2 \le \xi_0 + \xi_{\max} \times \frac{T_p}{2} \\ p_2 - \xi_{\max} \times \frac{T_p}{2} &\le p_2 \le p_2 + \xi_{\max} \times \frac{T_p}{2} \end{aligned} \tag{32}$$

From this group of control functions, a group of trajectories is generated as shown in fig. 6. By simply varying each parameter values, we obtain a wide variety of movement possibilities.

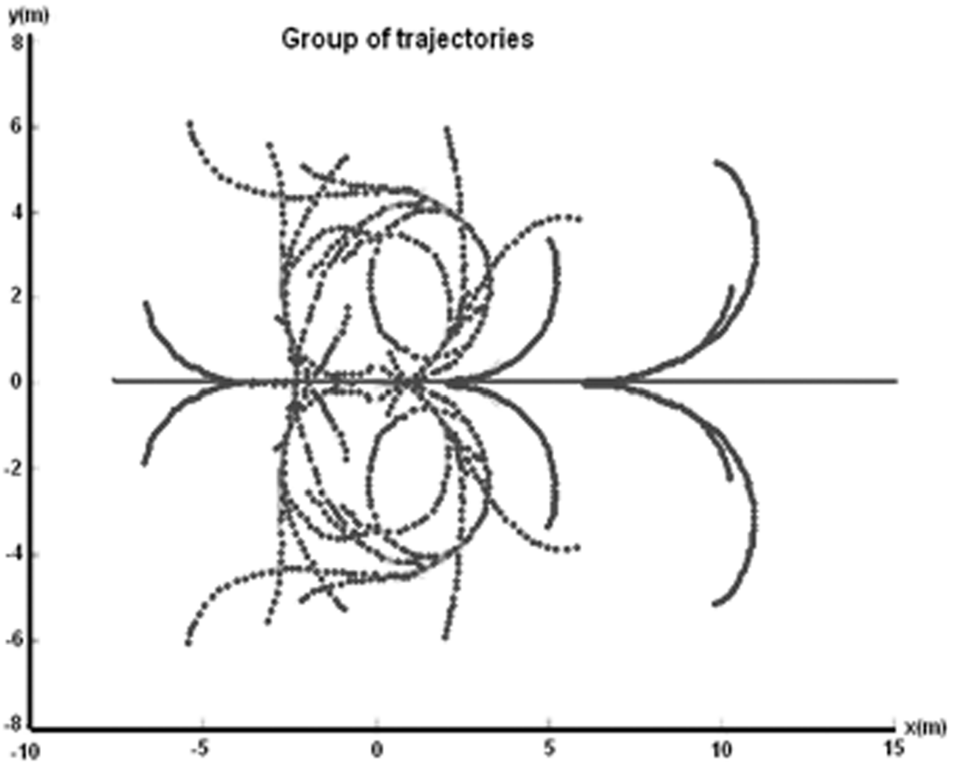


Figure 6: Group of trajectories

The Predictive control requires a linear kinematic model giving by (Bak et al, 2000). It is based on the following model, consisting for monitoring a straight line:

$$\dot{s} = \vartheta \cdot \cos\left(\theta - \psi\right) \tag{33}$$

$$\dot{d} = \vartheta \cdot \sin\left(\theta - \psi\right) \tag{34}$$

$$\dot{\theta} = \omega \tag{35}$$

Where is the distance to the next intersection. This model can be linearized around the operating point ($d = 0$, $\theta - \psi = \theta_e = 0$), this gives

$$\dot{d} = \upsilon \cdot \theta_e$$

$$\dot{\theta}_e = \omega \tag{36}$$

And can be discretized with a sampling period

$$d\left(k+1\right) = d\left(k\right) + h \cdot v \cdot \left[\theta\left(k\right) - \psi\left(k\right) + \frac{h}{2}\right]$$

$$\theta\left(k+1\right) = \theta\left(k\right) + h \cdot \omega\left(k\right) \tag{37}$$

The equations can be writing in state spaces representation

$$\begin{pmatrix} d\left(k+1\right) \\ \theta\left(k+1\right) \end{pmatrix} = \begin{pmatrix} 1hv \\ 01 \end{pmatrix} \begin{pmatrix} d\left(k\right) \\ \theta\left(k\right) \end{pmatrix} + \begin{pmatrix} \frac{h^2}{2} v \\ h \end{pmatrix} w\left(k\right) + \begin{pmatrix} 0 & hv \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ \psi\left(k\right) \end{pmatrix} \tag{38}$$

Orin compact form it is writing

$$z\left(k+1\right) = B \cdot z\left(k\right) + B_{\omega} \cdot \omega\left(k\right) + B_r \cdot r\left(k\right) \tag{39}$$

$$\begin{pmatrix} d\left(k\right) \\ \theta\left(k\right) \end{pmatrix} = z\left(k\right) : \text{System State} \tag{40}$$

$$\begin{pmatrix} 0 \\ \psi\left(k\right) \end{pmatrix} = r\left(k\right) : \text{The reference vector} \tag{41}$$

The criterion chosen to minimize the horizon is as following

$$J\left(k\right) = \lambda \cdot \omega^2\left(k+n\right) + \sum_{n=0}^{N} \left[\hat{z}\left(k+n\right) - r\left(k+n\right)\right]^T Q \left[\hat{z}\left(k+n\right) - r\left(k+n\right)\right] \tag{42}$$

Where $\hat{z}$ is the predicted output, Q is theWeighting matrix, λ is a scalar weight and N is the horizon.

Using Eq. (39)in Eq. (42), we have the following equation

$$J\left(k\right) = \left[\hat{Z}\left(k\right) - R\left(k\right)\right]^T IQ \left[\hat{Z}\left(k\right) - R\left(k\right)\right] + \lambda \Omega^T\left(k\right) \Omega\left(k\right) \tag{43}$$

Where

$$\hat{Z}(k)=\begin{pmatrix}\hat{z}(k/k)\\ \vdots\\ \hat{z}(k+N/k)\end{pmatrix}=Fz(k)+G_{\omega}\cdot\Omega(k)+G_r r(k) \tag{44}$$

$$I_Q=\begin{pmatrix}Q&0&0&\cdots&0\\ 0&Q&0&\cdots&0\\ \vdots&\ddots&\ddots&\ddots&\vdots\\ \vdots&\ddots&\ddots&Q&0\\ 0&\cdots&\cdots&0&Q\end{pmatrix} \tag{45}$$

Where

$$\Omega(k)=\left(\omega(k)\cdots\omega(k+N)\right)^T \tag{46}$$

$$R(k)=\left(r(k)\cdots r(k+N)\right)^T$$

$$=\begin{pmatrix}0\\ \psi(k)\end{pmatrix}^T=\begin{pmatrix}0&\cdots&0\\ \psi(k)&\cdots&\psi(k+N)\end{pmatrix}^T \tag{47}$$

And

$$G_i=\begin{pmatrix}0&0&\cdots&0&0\\ B_i&0&\cdots&0&0\\ AB_i&B_i&\cdots&\vdots&\vdots\\ \vdots&\ddots&\ddots&0&0\\ A^{N-1}B_i&\cdots&AB_i&B_i&0\end{pmatrix} \tag{48}$$

$$F=(I\,A\,\ldots.\,A^N)^T \tag{49}$$

4.3.4. The predictive algorithm

The goal is to find the control sequence ω(*k*); $1\le K\le n$ that minimizes *J*(*k*).

$$\Omega(k)=-K_z\cdot z(k)-L_r\cdot R(k) \tag{50}$$

Where

$$L_z=\left(\lambda+G_{\omega}^T I_Q G_{\omega}\right)^{-1}G_{\omega}^T I_Q G_{\omega}F$$

$$L_r=\left(\lambda+G_{\omega}^T I_Q G_{\omega}\right)^{-1}G_{\omega}^T I_Q\left(G_r-1\right) \tag{51}$$

In fact only the next order ω(*k*) is applied to the system. Therefore only the first line of L_z and the two first line of L_r. That gives the following

$$\omega(k)=-k_{\theta}\cdot\theta(k)-k_d\cdot d(k)-K_{\psi}\cdot\Psi(k) \tag{52}$$

Where k_d and k_θ are scalar and vector k_ψ as $[1 \times (N + 1)]$ ψ multiplying the reference vector $\psi(k)$.

In the case of a turn (sequence of two lines), the vector reference has the following form

$$\psi(k) = \left(\left(\psi_1 \cdots \psi_1, \psi_2 \cdots \psi_2 \right) \right)^T \tag{53}$$

5. IMPLEMENTATION AND ROBOT CONSTRUCTION

5.1. Architecture

A Raspberry Pi module will contain data processing. An application will play the role of a planner. Indeed it's the path provider, at first it will provide several consecutive points that the robot must cross to reach its final goal. The architecture is shown in Fig. 7.

The synthesis of the two regulators in cascade was made and discussed. We must now put it in common and do all the required adaptations.

The block diagram in Fig. 8 shows the general principle of the regulation.

The block "strategy" is the artificial intelligence of the robot. It dictates the way and the desired velocity.A simulation of this control principle has been made on the softwareMatlab Simulink simulation. This optimizes some parametersto save time on the actual implementation.

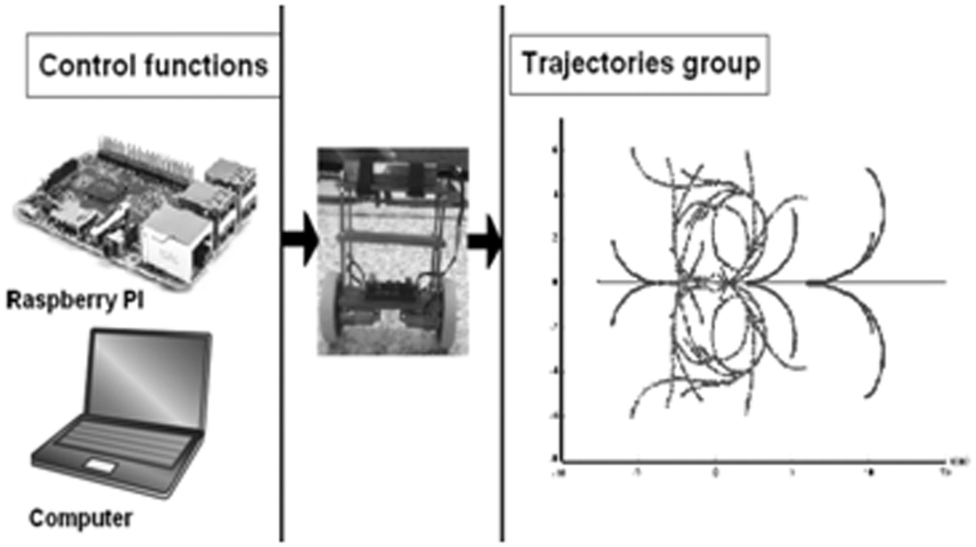


Figure 7: System Architecture

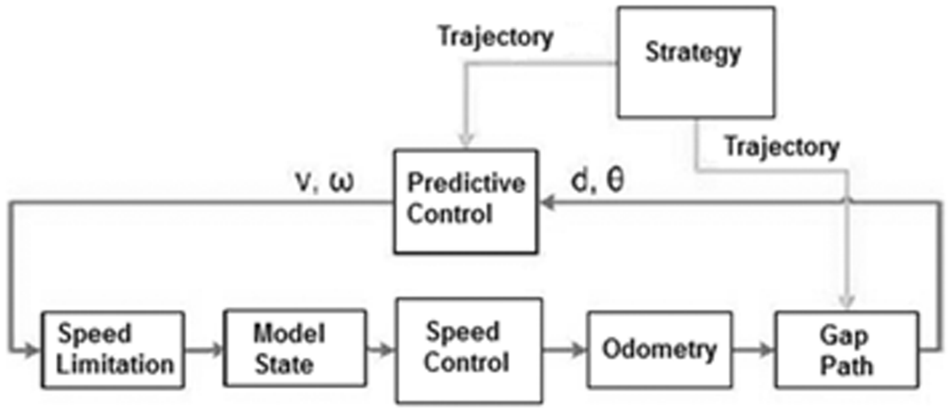
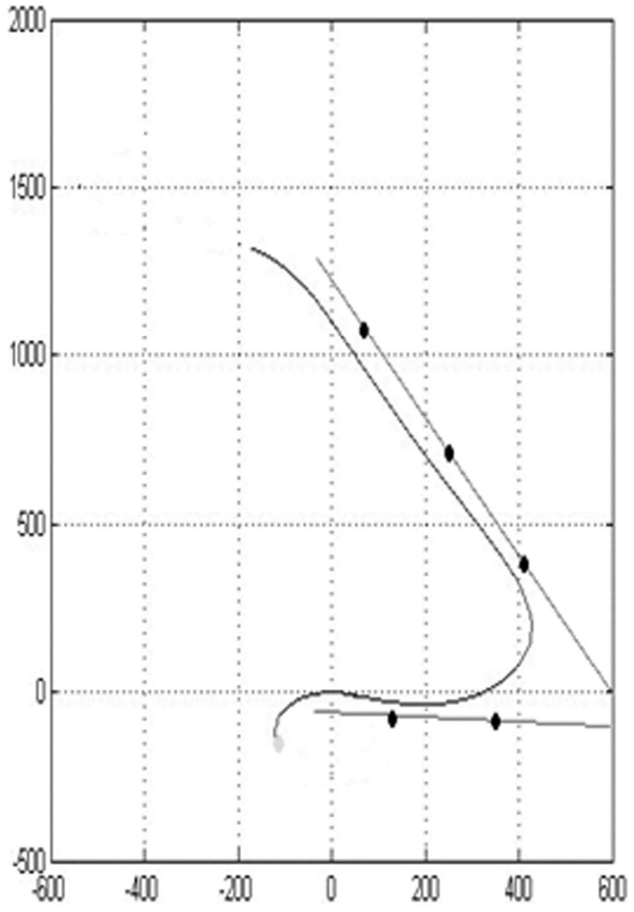
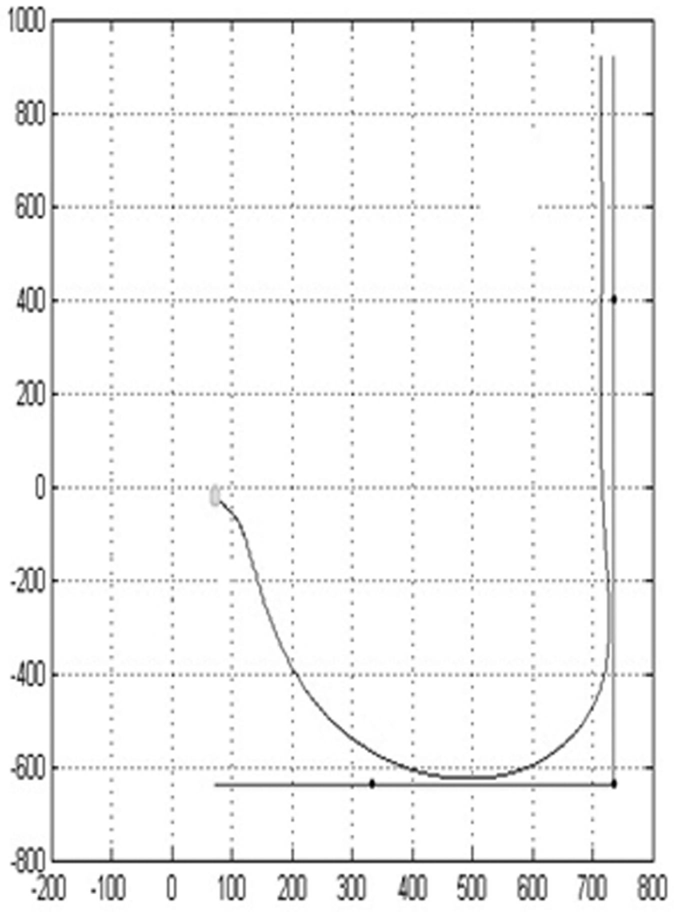


Figure 8: General functional regulation



(a) Simulation for 60° angle



(b) Simulation for 90° angle

Figure 9: Simulation Results

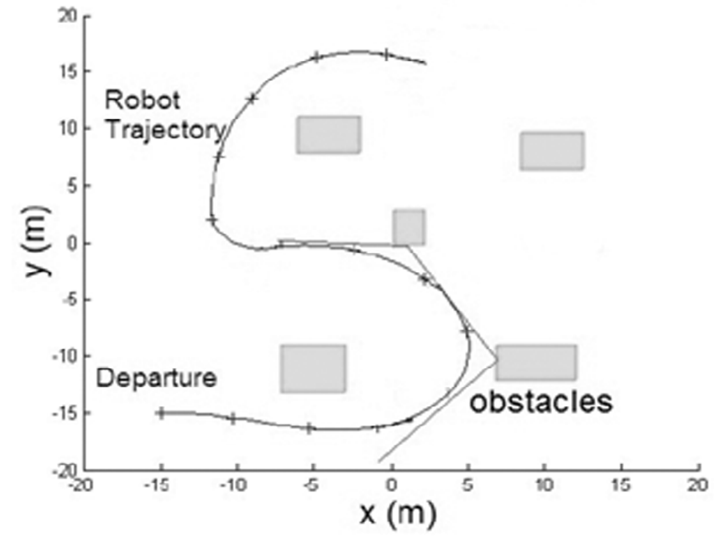


Figure 10: Simulation Results for obstacles avoidance

The best results of simulationsare shown in Figs. 9 (a)-(b). We see the reference red path line and how far the robot follows the line.It is noted that even with an initial position outside path (the angle is also set to a divergent path), the robot huntalthough the way.

Figure 10 shows a simulation for robot navigation moving surrounded by obstacles. The position of obstacles on a map of grid occupancy size 40 * 40 m ², they are represented by green areas, and are considered expanded to take into account the size of the robot. The robot path is represented by the blue line, each small cross marking a new iteration of the navigation algorithm (period Te).

The following simulation shows the behavior of the robot in the case where the trajectory of Reference cross an obstacle. These simulations show how the developed navigator allows to adapt the robot path to bypass the obstacles, while following as possible the reference trajectory.

We used the simulated algorithm, which as we have previously shown to be effective to find the paths in this type of situation. In the figures, the track of travel performed by the robot is shown in blue. Therefore, the trace shown between two cross corresponds to the movements made by the robot during a sampling period.

5.2. Sensors

The sensors are responsible for returning the state of movement of the robot. The system, based on these information, will guide the robot and control the motion. There are two sensors in our robot gyroscope and accelerometer.

The accelerometers are responsible for calculating the unstable robot tilting as it faces a free fall. In our case, acceleration is calculated on two axes so that we can determine the tilt angle.

The gyroscope is combined to the accelerometer to correct the angle and measure the rate of one direction.

5.3. Hardware

The hardware specific to the robot are shown in Fig. 10.

Arduino is an ATmega1284P surface mount breadboard containing the control motors for speed and displacement [26].

Its microcontroller was very powerful to receive data from the raspberry PI module which trace the idealtrajectory. We developed the program on its "Arduino IDE" software.

The Devantech RD01 Drive is a specific Robot Drive system, containing an MD23 motor drive module, two EG30 motors with encoders and two 120mm wheels with pivots already fitted.

We used a 12V battery to power the combined system. The engine is composed of the Arduino, the motors with encoders and the sensors (accelerometer and gyroscope).

Several tests on hardware were used to adjust several values and put the parameters of the theoretical calculations to practice.

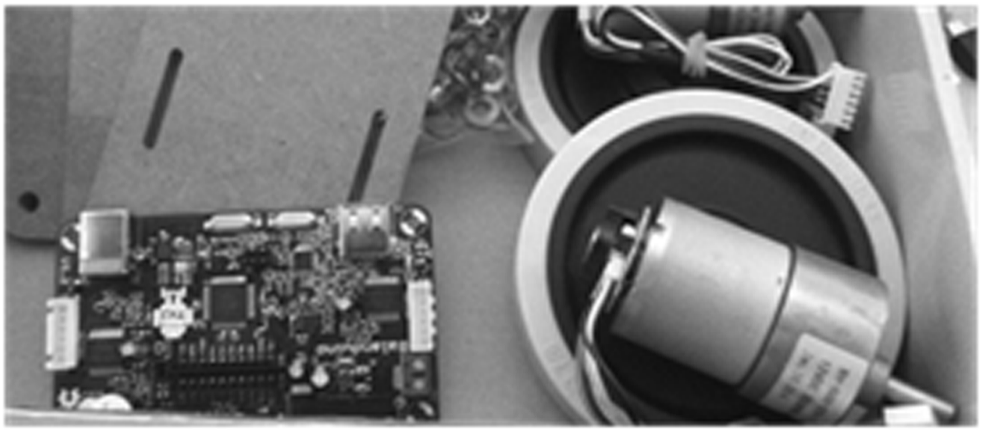


Figure 11: Components of the Robot

The MEMS (microelectromechanical systems) chip of the 6-axis MPU-6050-gyroscope-accelerometer is very accurate with an analog-digital conversion of 16 bits simultaneously on each connection and an I2C interface. The reading of this sensor measurement is easy.

The sensor contains a FIFO register of 1024 bytes that the Arduino microcontroller can read, being informed by an interrupt signal. The module operates as a slave on the I2C bus with respect to the Arduino (SDA pins, SLC) but can also control another downstream device with AUX and AUX-DA-CL. Its consumption is low, 3.9 mA with 6 activated sensors.

The sensor has a DMP (Digital Motion Processor) able to do quick calculations directly on the chip from the raw sensor measurements but it is very slow. It is therefore easier to process raw measurements on the Arduino board.The assembled robot is shown in fig. 11.

5.4. Software Implementation

After having assembled the different hardware and sensors, we need to know how to exploit the different information. The gyroscope provides a measure of the angular velocity. By integrating this measure, we can get the value of the pivot angle. However, this operation will integrate the noise of the system itself, accumulating in a few iterations, a large drift. The accelerometer measures static and dynamic acceleration. In order to separate the two components, we are forced to use a low pass filter (static component: gravity) and a high pass filter (dynamic acceleration).

Once separated, a simple trigonometric operation, we can calculate the slope. However, the passage through the filters of this type (average values) causes a significant delay in the response making the system unstable.

As we have a gyro drift more and more important over time and a slow and noisy accelerometer as inputs in our system. The solution is to “merge” (filter) the two readings.

The solution was to choose the Kalman filter shown in fig. 12. It is a filter with an infinite impulse response that estimates the state of a dynamic system from a series of incomplete and noisy measurements[27].

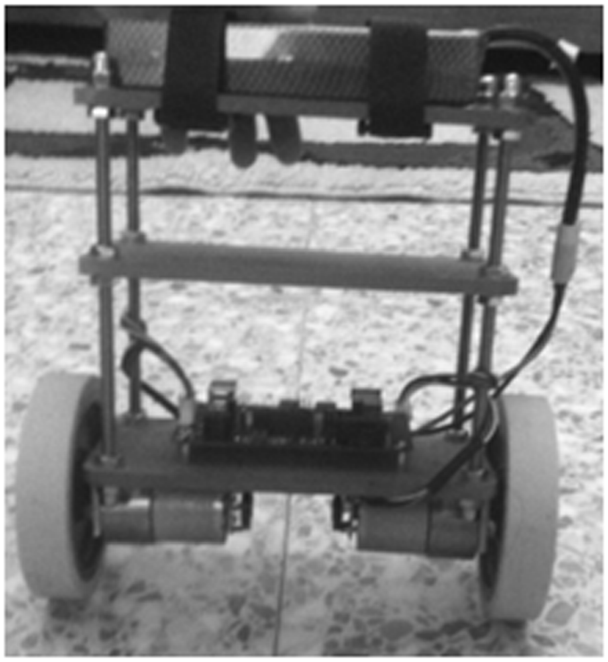


Figure 12: Assembled parts of the Robot

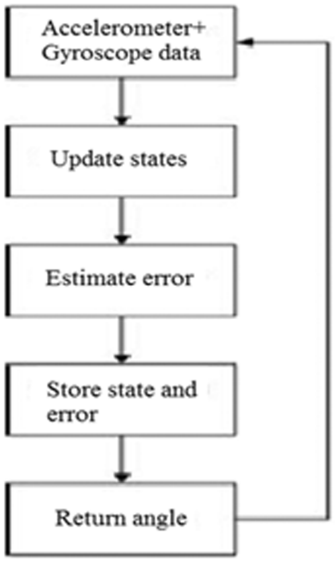


Figure 13: Kalman Process

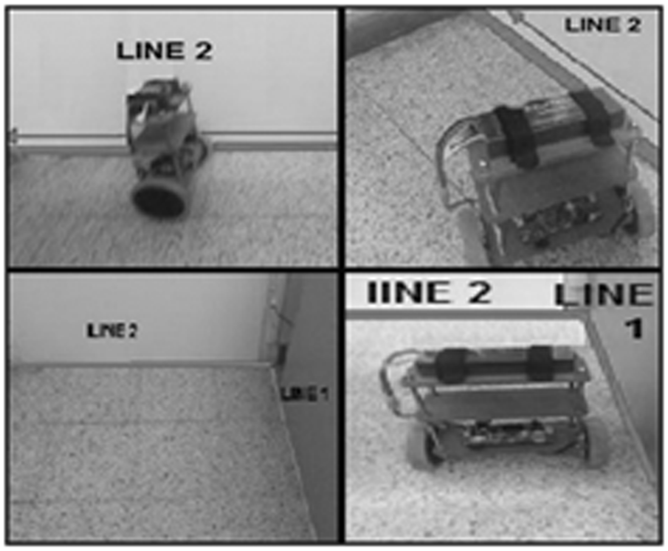


Figure 14: The robot following the path and responding to constraints

The Raspberry will not be directly used, connected to a screen and a keyboard for example. As part of a server, the Raspberry will be permanently connected and operate without any interruption. It will remain close to the internet box and will be controlled remotely. With this in mind, we will seek the stability of the system, which means the use of a Raspbian distribution. In general, the server requests a capacity to handle a high scalability (that is to say, a significant increase in the number of resources). This means the server requires an SD card allowing rapid writing, and a Raspberry with more RAM. The Raspberry PI will play the role of an onboard computer and will allow operation of the Matlab program which will give the path to be followed.Here are the experimental results on the Robot in Fig. 13.

The predictive control shows its effect, the robot anticipates the turn and reach the second line. A modification has been made compared to theory to get the right result shown previously. we added an anticipation of 80 mm. it corresponds in fact to the horizon change of the robot, which must react sooner. He therefore believed 80 mm closer to the tipping point. this allows avoid any overshoot. Simulink diagram was used for the simulation before being implemented on the Robot. It is the general scheme, it has many sub parts.

6. CONCLUSION

The purpose of this work was to control a robot on a path. The robot wascontrolled through state method and on a path according an improved control law. These two controllers in cascade made it possible to follow a path. A simulation allowed to find the correct settings required forimplementation on the real system. The simulations and implementation validate the effectiveness of our predictive control in crowded map. The trajectories calculated from the direct kinematic model of the robot and respecting its own kinematic constraints are generated. It allows the robot to bypass the presenting obstacles in its path. This Control thus ensures local adaptation to the environment and the robot contraints and to the path provided by the planner.

ACKNOWLEDGMENTS

The research described in this paper was financially supported by the LTSIRS Laboratory, National Engineering School of Tunis, Tunisia.

APPENDIX

Notation

$F_i(t)$	Friction force of the wheel on the ground
$i_i(t)$	Power in the motor
I_i	Moment of inertia relative to the wheel
I_0	Moment of inertia of the robot
k_i	Motor constant
$M_i^1(t)$	Motor torque attached to the wheel
$M_i^2(t)$	Friction torque of the wheel n i
n_i	Reduction factor
$M_i^3(t)$	Torque caused by the wheel's dynamic
m_0	Mass of the robot
R_i	Motor resistance
r_i	Radius of the
R_i	Motor resistance
r_i	Radius of the wheel
r_0	Half width between the wheels
$U_i(t)$	Voltage applied to the motor
$x(t)$	Linear position of the robot
θ_0	Angular position of the robot

Appendix 2

$$a_r^r = -\frac{\left(4I_l r_0^2 + r_1^2\left(I_0 + m_0 r_0^2\right)\right)\left(k_r^2 n_r^2 + f_r R_r\right)}{I_0 I_r r_l^2 + I_0\left(I_l + m_0 r_l^2\right) r_r^2 + r_r^2\left(4I_l I_r + I_r m_0 r_r^2\right) R_r}$$

$$a_l^r = -\frac{k_l\, n_l\, n_r\left(I_0 - m_0 r_0^2\right) r_l}{I_0 I_l r_l^2 + I_0\left(I_l + m_0 r_l^2\right) r_l^2 + r_l^2\left(4I_r I_l + I_l m_0 r_l^2\right) R_l}$$

$$a_r^l = -\frac{k_r\, n_r\, n_l\left(I_0 - m_0 r_0^2\right) r_r}{I_0 I_r r_l^2 + I_0\left(I_l + m_0 r_l^2\right) r_r^2 + r_r^2\left(4I_r I_r + I_r m_0 r_r^2\right) R_r}$$

$$a_l^l = -\frac{\left(4I_r r_0^2 + r_r^2\left(I_0 + m_0 r_0^2\right)\right)\left(k_l^2 n_l^2 + f_l R_l\right)}{I_0 I_l r_r^2 + I_0\left(I_r + m_0 r_r^2\right) r_l^2 + r_l^2\left(4I_r I_l + I_l m_0 r_l^2\right) R_l}$$

$$b_r^r = -\frac{k_r n_r \left(4I_l r_0^2 + r_l^2 \left(I_0 + m_0 r_0^2\right)\right)}{I_0 I_r r_l^2 + I_0 \left(I_r + m_0 r_r^2\right) r_r^2 + r_r^2 \left(4I_l I_r + I_r m_0 r_r^2\right) R_r}$$

$$b_l^r = -\frac{k_l n_l r_r r_l \left(I_0 - m_0 r_0^2\right)}{I_0 I_l r_r^2 + I_0 \left(I_r + m_0 r_r^2\right) r_l^2 + r_l^2 \left(4I_r I_l + I_l m_0 r_l^2\right) R_l}$$

$$b_r^l = -\frac{k_r n_r r_l r_r \left(I_0 - m_0 r_0^2\right)}{I_0 I_r r_l^2 + I_0 \left(I_l + m_0 r_l^2\right) r_r^2 + r_r^2 \left(4I_l I_r + I_r m_0 r_r^2\right) R_r}$$

$$b_l^l = -\frac{k_r n_r \left(4I_l r_0^2 + r_l^2 \left(I_0 + m_0 r_0^2\right)\right)}{I_0 I_r r_l^2 + I_0 \left(I_r + m_0 r_r^2\right) r_r^2 + r_r^2 \left(4I_l I_r + I_r m_0 r_r^2\right) R_r}$$

REFERENCES

[1] F. Grasser, A. D'arrigo, S. Colombi andA.C. Rufer, "JOE: a mobile inverted pendulum," *IEEE Transactions on Industrial Electronics*, **49** (1), 107-114, 2002.

[2] R.C. Ooi, *Balancing a Two-Wheeled Autonomous Robot*, Final Year Thesis,School of Mechanical Engineering, University of WesternAustralia, Australia, 2003.

[3] K.C.R. Ho, *Balancing Wheeled Robot*, Research Project, University of Southern Queensland, Australia, 2005.

[4] R. Grepl, "Balancing wheeled robot: effective modelling, sensory processing and simplified control," *Engineering Mechanics*, **16** (2), 141–154, 2009.

[5] Y. Takita, H. Dateand H. Shimazu, "Competition of two-wheel inverted pendulum type robot vehicle on MCR Course," The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems,5578-5584, 2009.

[6] Y.O. Chee and M.S.Z. Abidin,"Design and development of two wheeled autonomous balancing robot," IEEE 4thStudent Conference on Research and Development, Selangor, 169-172, 2006.

[7] S.W. Nawawi, M. Ahmad and J.H.S. Osman, "Real-time control of a two-wheeled inverted pendulum mobile robot," World Academy of Science, Engineering and Technology,**39**, 214-220, 2008.

[8] C. Tsai , C. Chan and Y.H. Fan, "Planned navigation of a self-balancing autonomous service robot," *IEEE International Conference on Advanced Robotics and Its Social Impacts*, Taipei, 1-6**,** 2008.

[9] J.C. Latombe,*Robot Motion Planning*, Kluwer Academic Publishers, Boston, 1991.

[10] H. Choset,*Sensor Based Motion Planning: the Hierarchical Generalized Voronoi Graph*, PhDThesis, California Institure of Technology, 1996.

[11] B. Chazelle and L.J. Guibas, "Visibility and intersection problems in plane geometry" , *Discrete and Computational Geometry*, **4**, 551-581, 1989.

[12] K. Shin and N. McKay,"A dynamic programming approach to trajectory planning of the robotic manipulators", *IEEE Transactions on Automatic Control*, **31** (6), 491-500, 1996.

[13] P.E. Hart, "A formal basis for the heuristic determination of minimum cost paths,"*IEEE Transactions on Systems Science and Cybernetics*, **4** (2), 100-107, 1986.

[14] A. Stentz, "Optimal and efficient path planning for partially-known environments,"*Proceedings of the IEEE International Conference on Robotics and Automation*, San Diego, 3310-3317, 1994.

[15] M. Defoort, A. Kokosy, T. Floquet, W. Perruquetti and J. Palos, "Motion planning for cooperative unicycle-type mobile robots with limited sensing ranges: A distributed receding horizon approach,"*Robotics and Autonomous Systems,* **57** (11), 1094-1106, 2009.

[16] I. Skrjanc and S. Blazio,"Predictive functional control based on fuzzy model: design and stability study", *Journal of Intelligent &Robotic Systems*,**43**, 283-299, 2005.

[17] G. Klancar and I. Skrjanc, "A case study of the collision-avoidance problem based on Bernstein-Bézier path tracking for multiple robots with known constraints*,"Journal of Intelligent &Robotic Systems*, **60** (2), 317-337, 2010.

[18] A. Micaelli and C. Samson, “Trajectory tracking for unicycle-type and two-steering-wheels mobile robots,” [Research Report] RR-2097, INRIA, 1993.

[19] J.S. Hu, M.C. Tsai, F.R. Hu and Y. Hori, “Robust control for coaxýal two-wheeled electrýc vehýcle,” *Journal of Marine Science and Technology*,**18** (2),172-180, 2010.

[20] G. Chi, J. Hausbach and B. Hunter, *Segbo*, Senior Design Project, University of Illinois at Urbana-Champaign, USA, 2005.

[21] H.G. Bock and K. Plitt, “A multiple shooting algorithm for direct solution of optimal control problems,” 9th IFAC World Congress, Budapest, Pergamon Press, 1984.

[22] G. Klancar and I. Skrjanc, “Tracking-error model-based predictive control for mobile robots in real time,” *Robotics and Autonomous Systems*,55 (6), 460-469, 2007.

[23] E. Courtial, *Commande predictive et estimation d’état de systèmes non linéaires*, Rapport de thèse,Université Claude Bernard – Lyon, 1996.

[24] T. Vallius and J. Röning, “Embedded object concept: case balancing two-wheeled robot,” Proceedings of the SPIE, **6764**, 2007.

[25] M. Bak, K. Poulsen and O. Ravn, “Path following mobile robot in the presence of velocity constraints,” Kongens Lyngby, Denmark : Technical University of Denmark, 2000.

[26] S. Vaidyanathan and K. Rajagopal, “Analysis, control, synchronization and LabVIEW implementation of a seven-term novel chaotic system,” *International Journal of Control Theory and Applications*, **9** (1), 151-174, 2016.

[27] S. Vaidyanathan and A. Boulkroune, “A novel hyperchaotic system with two quadratic nonlinearities, its analysis and synchronization via integral sliding mode control,” *International Journal of Control Theory and Applications*, **9**(1), 321-337, 2016.

[28] S. Vaidyanathan, “Mathematical analysis, adaptive control and synchronization of a ten-term novel three-scroll chaotic system with four quadratic nonlinearities,” *International Journal of Control Theory and Applications*, **9**(1), 1-20, 2016.