

## International Journal of Control Theory and Applications

ISSN : 0974-5572

© International Science Press

Volume 10 • Number 34 • 2017

### Vision Control Algorithm for Two-wheeled Inverted Pendulum Mobile Robot Navigation

Sabri M. Ben Mansour<sup>\*1</sup>, Sami Cherif<sup>1</sup>, Sundarapandian Vaidyanathan<sup>2</sup> and  
Saber M. Naceur<sup>1</sup>

<sup>1</sup>LTSIRS, Remote Sensing and Spatially Referenced Information Systems Laboratory, National Engineering School of Tunis, Tunisia  
E-mail: sabri.benmansour@gmail.com

<sup>2</sup>Research and Development Centre, Vel Tech University, Chennai-600 062, Tamil Nadu, India

**Abstract:** This paper addresses Visio and path following control problem of a nonholonomic Two-Wheeled Inverted Pendulum Mobile Robot. We propose control architecture based on two control layers. A speed inner loop control scheme is first designed based on state feedback technique to ensure stability of the inverted structure of the robot. A second outer loop control scheme is proposed to help the robot navigate along a desired path formed by a set of way points. It is designed inspiring the model predictive control 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. The obstacle detection architecture based on the attributes of color and texture has been developed to be implemented on an Raspberry PI and is designed as a generic high-speed image processing device. The optimization criteria are based on a maximization of performance in terms of image processing per second and a minimization of consumed resources. Our obstacles detection algorithm consists of three main steps: the color transformation, the calculation of the color and texture attributes and their classification.

**Keywords:** Mobile robot, navigation, stability, Predictive control, Obstacles detections, Image processing

#### 1. INTRODUCTION

The problem of non-holonomic systems control has attracted numerous investigations in the past. A thorough studied case, with great practical significance, is the wheeled mobile robot with a kinematic model similar to a unicycle [1,2]. The differentially driven mobile robots that are very common in practical applications also have the same kinematic model. Although many researchers coped with the more difficult problem of stabilizing dynamic models for different types of mobile robots [3-6], the basic limitations of mobile robot control still come from their kinematic model. Kinematic control laws are also very important from the practical point of view, since the wheel-velocity control is often implemented locally on simple micro-controller based hardware.

Traditionally, the problem of mobile robot control has been approached by stabilization point or by redefining the problem as a tracking control one. There are also some approaches that tackle both problems simultaneously. We believe that the tracking control approach is somewhat more appropriate, since the non-holonomic constraints and other control goals (obstacle avoidance, minimum travel time, and minimum fuel consumption) are implicitly included in the path-planning procedure [7,8]. We were looking for to develop certain tools, as the paths optimization methods, and adapt them to our problem navigation. The aim was to get an under constraints optimization algorithm which can be implemented in a real-time application. Some recent control methods are discussed in [26-30].

The navigation method that we propose follows several steps: to enable the robot to join the path provided by the modelling environment, a catch curve is generated with the authors approach in [9].Based on the knowledge of the model of the robot 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.

We present an approach for obstacles detection of a scene from two independent methods processed in parallel. The first method calculates the attributes of color and texture from a single color image. In this method the color attributes are calculated using the color space CIE-Lab, because of its high immunity to illumination change. The chromatic components a and b of this space are used as color attributes in the classification, while the brightness  $L^*$  is used to calculate the texture attributes as it provides the greatest contrast. A collection of images is required to create a database with the color and texture attributes. A supervised learning technique was evaluated: The Adaboost algorithm which has the best performance for detecting the ground.

The problem statement is given in Section 2. The algorithm for obstacles detection is described in section 3. The kinematic model and the control design are calculated in section 4. The speed and trajectory control are developed in section 5. In Section 6, implementation and robot construction is discussed. The conclusion are stated in Section 7.

## 2. PROBLEM STATEMENT

Considering a differentially driven, two-wheeled mobile robot like the one showed in Figure 1, where  $(x, y)$  is the wheel-axis-centre position and  $\theta$  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 non holonomic constraint of the form

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

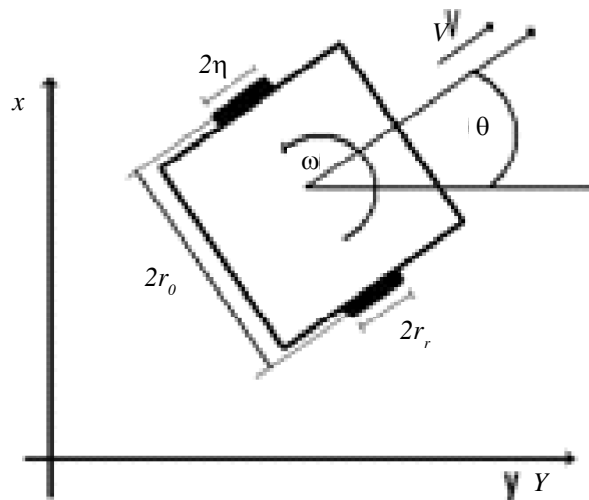


Figure 1: Mobile inverted pendulum

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

Where  $\dot{\theta}_r$  and  $\dot{\theta}_l$  are the angular velocity of the right wheel and the angular velocity of the left wheel.

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

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

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

where

$$e(k) = \theta(k) - \theta(k-1) \quad (6)$$

$$\Delta\theta_r = \theta_r(k) - \theta_r(k-1) \quad (7)$$

$$\Delta\theta_l = \theta_l(k) - \theta_l(k-1) \quad (8)$$

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 turn with a radius of curvature, to be rotatable about an obstacle.

### 3. ALGORITHM FOR OBSTACLES DETECTIONS

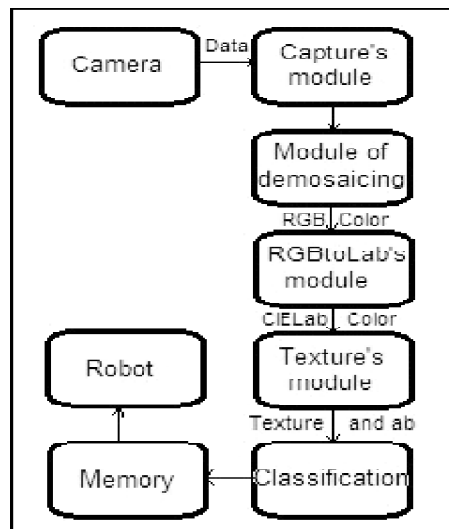


Figure 2: Algorithm diagram for the obstacles detection

Figure 2 shows the architecture designed to carry out the detection of obstacles. The first module is the image capture algorithm which provides the interface between the camera and the architecture. This module provides the raw data from the sensor to the demosaicing unit responsible for recovering the color image. This module outputs a data bus which transmits each pixel of the image in the RGB color space. Each color component of a pixel is coded on nb bits. The output of demosaicing module is linked to the three main modules of the architecture.

### 3.1. Color space transformation

The CIELab color transformation is a function of primary coordinates XYZ with Standard illuminant D65. Primary coordinates XYZ are calculated by a linear transformation from the RGB tri-stimulus components. It is obtained through the following matrix:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \quad (9)$$

The brightness  $L^*$  and the chromaticity components  $a$  and  $b$  are given by the following equations depending upon XYZ coordinates:

$$L^* = 116 \left[ f\left(\frac{Y}{Y_w}\right) \right] - 16 \quad (10)$$

$$a = 500 \left[ f\left(\frac{X}{X_w}\right) - f\left(\frac{Y}{Y_w}\right) \right] \quad (11)$$

$$b = 200 \left[ f\left(\frac{Y}{Y_w}\right) - f\left(\frac{Z}{Z_w}\right) \right] \quad (12)$$

where  $X_w$ ,  $Y_w$  and  $Z_w$  symbolize the white reference in the standard illuminant D65, and set at (0.950456, 1.000000, 1.088754),  $f(t)$  will manage the stability (or noise) of this Low-intensity space; it is defined as follows:

$$f(t) = \begin{cases} t^{\frac{1}{3}} \text{sit} > \left(\frac{6}{29}\right)^3 \\ \frac{1}{3} \left(\frac{29}{6}\right)^2 t + \frac{4}{29} \text{sit} < \left(\frac{6}{29}\right)^3 \end{cases} \quad (13)$$

### 3.2. Architecture for texture analysis

The architecture for the texture analysis consist of two main processes: sum and difference of images, and calculating texture attributes. These processes are shown inside two rectangles in the Fig. 11. The rectangle at the top represents the modulus of the sum and difference of histograms, the bottom represents the textural attributes itself divided into six modules. Five of them represent the first five texture attributes in M.Unser method. The sixth module is necessary to calculate the texture attributes of the variance and correlation.

The sum and difference of images is the first process executed during the analysis of dense texture. Addition and subtraction are the operations performed between the current pixel and the displaced pixels. Sixteen movements in total are used for four distances and four orientations, such as (1, 2, 3 and 4 pixels) and (45°, 90°, 135° and 180°) respectively. For each shifting, a processing window of  $K * L$  17 15 pixels is used to calculate the texture

attributes: average :  $\frac{1}{2} \sum_i i \cdot \widehat{P}_S(i)$ , variance:  $\frac{1}{2} (\sum_i (i - 2\mu)^2 \cdot \widehat{P}_S(i) + \sum_j j^2 \cdot \widehat{P}_D(j))$ , uniformity:  $\sum_j \frac{1}{1+j} \cdot \widehat{P}_D(j)$  contrast :  $\sum_j j^2 \cdot \widehat{P}_D(j)$  and correlation:  $\frac{1}{2} (\sum_i (i - 2\mu)^2 \cdot \widehat{P}_S(i) - \sum_j j^2 \cdot \widehat{P}_D(j))$ .

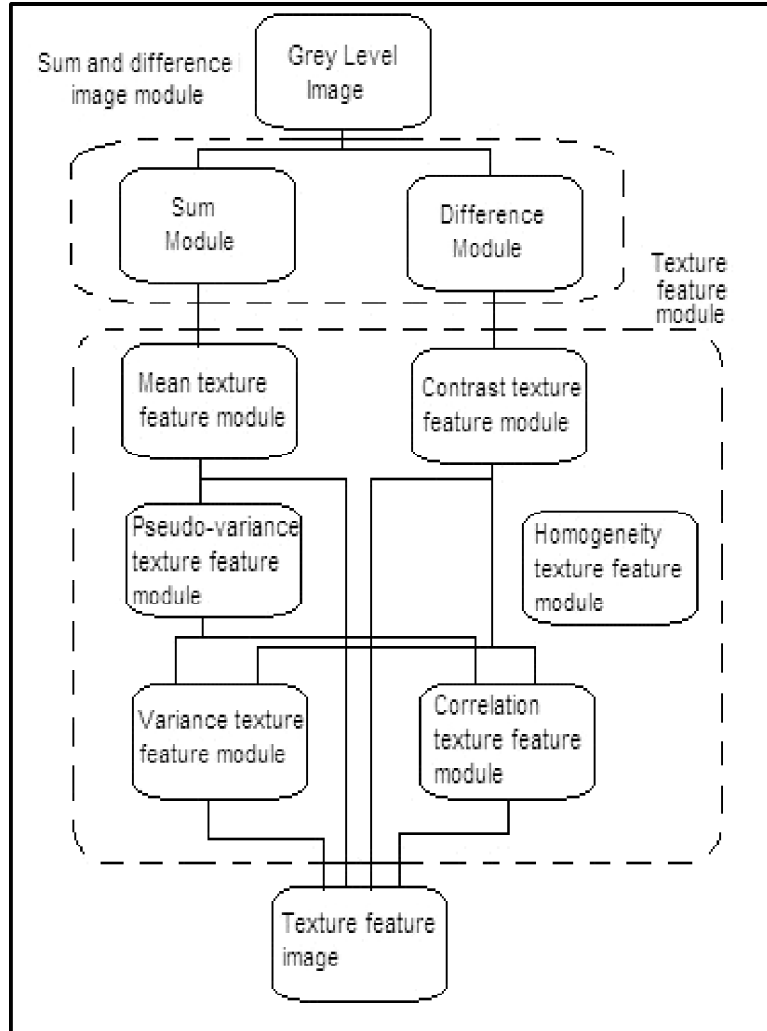


Figure 3: Architecture diagram for texture analysis based on sums and differences of histograms

### 3.3. Architecture for the classification algorithm

The database was built in a supervised mode which required the prior determination of the relevant object classes for the intended application. The number of classes is fixed at only two because it is the detection of a particular object in the scene and not the identification of each of the objects. The database was created from sample images of some obstacles and some parts of the ground. One file contains the samples in the space of the color attributes and texture calculated from the base image. Each sample is marked with a label indicating whether it belongs to the class of interest (soil) or not (obstacles).

### 3.4. Experimental results

The images were acquired with tested digital camera STMicroelectronics. Obstacle detection was performed by using the results of evaluations in the previous section. The classification method used is the discrete AdaBoost

algorithm with 20 weak assumptions. Modeling soil was made using 22 attributes of color and texture. The final method for the detection of obstacles was implemented on an embedded architecture. Thus, we chose to make dense obstacle detection (soil in blue and obstacles in red), so the treatment is done pixel by pixel. Results are in Figure 4 (a) and (b).



Figure 4 (a): Original images

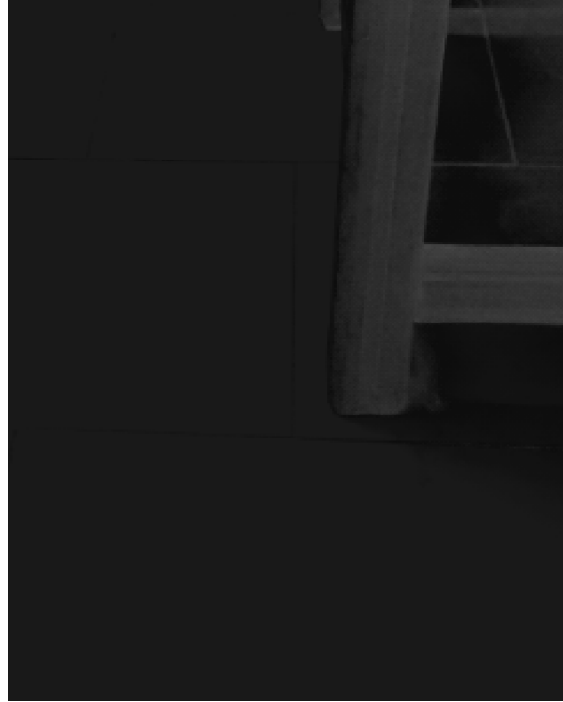


Figure 4 (b): images resulting from the classification

#### 4. THE KINEMATIC MODEL AND THE CONTROL DESIGN

It is possible to model the system described above by the Newton-Euler method.

##### 4.1. TWIPMR modeling

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.

$$\begin{aligned} \dot{X} &= A \cdot X + B \cdot U \\ Y &= C \cdot X \end{aligned} \quad (14)$$

Then we get the explicit equations

$$\begin{pmatrix} \dot{\theta}_r \\ \ddot{\theta}_r \\ \dot{\theta}_l \\ \ddot{\theta}_l \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & a_r^r & 0 & a_l^r \\ 0 & 0 & 0 & 1 \\ 0 & a_r^l & 0 & a_l^l \end{pmatrix} \cdot \begin{pmatrix} \theta_r \\ \dot{\theta}_r \\ \theta_l \\ \dot{\theta}_l \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ b_r^r & b_l^r \\ 0 & 0 \\ b_r^l & b_l^l \end{pmatrix} \cdot \begin{pmatrix} u_r \\ u_l \end{pmatrix} \quad (15)$$

$$\begin{pmatrix} \theta_r \\ \theta_l \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \theta_r \\ \dot{\theta}_r \\ \theta_l \\ \dot{\theta}_l \end{pmatrix} \quad (16)$$

where

$\theta_r$  : The angular position of the right wheel

$\theta_l$  : The angular position of the left wheel

$\dot{\theta}_r$  : The angular velocity of the right wheel

$\dot{\theta}_l$  : The angular velocity of the left wheel

The details for Eqs (15) and (16) 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.

$$\begin{aligned} X(k+1) &= \Phi \cdot X(k) + \Gamma \cdot U(k) \\ Y(k) &= C \cdot X(k) \end{aligned} \quad (17)$$

Where  $\Phi$ ,  $\Gamma$ ,  $C$  and  $D$  are matrices giving by Matlab calculation.

## 4.2. Control design

In the literature, the control law for the tracking path is based on the assumption that the robot speed limits are directly transmitted [10] [11]. This hypothesis corresponds to a control in perfect speed.

This separation between low-level (robot speed control) and High-level (tracking path) in Figure 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.

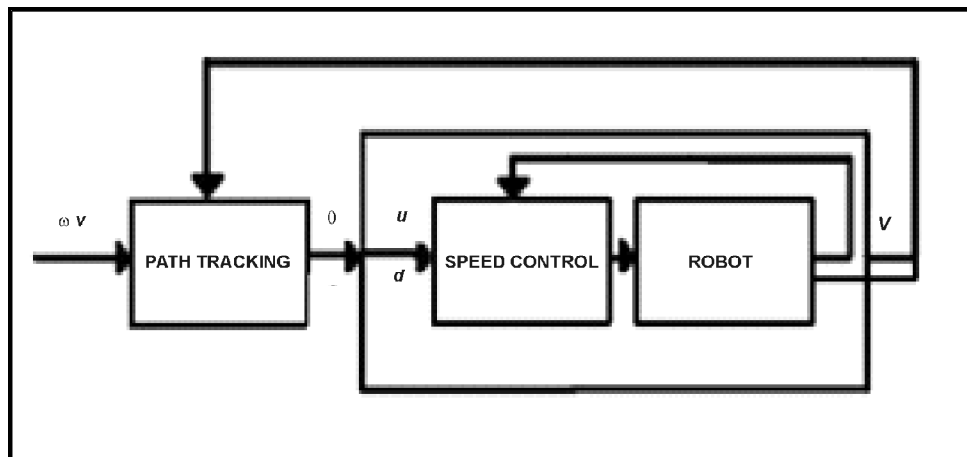


Figure 5: The two controllers in cascade

## 5. THE SPEED AND TRAJECTORY CONTROL

### 5.1. 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:

$$J = \frac{1}{2} \sum_{k=0}^N [x^T(k) \cdot Q_1 x(k) + u^T(k) \cdot Q_2 u(k)] \quad (19)$$

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.

$$Q_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 100'000 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100'000 \end{pmatrix} \quad (20)$$

$$Q_2 = \begin{pmatrix} 10 & 0 \\ 0 & 10 \end{pmatrix} \quad (21)$$

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

$$U(k) = K_G \cdot X(k) \quad (22)$$

Where  $K_G$  contains the gain obtained by Matlab calculation

## 5.2. 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, but that can take into account certain geometric constraints robot (size, turning radius). Angular velocity  $\omega = \dot{\theta}$  of the robot is used as a control variable as shown in Figure 4.

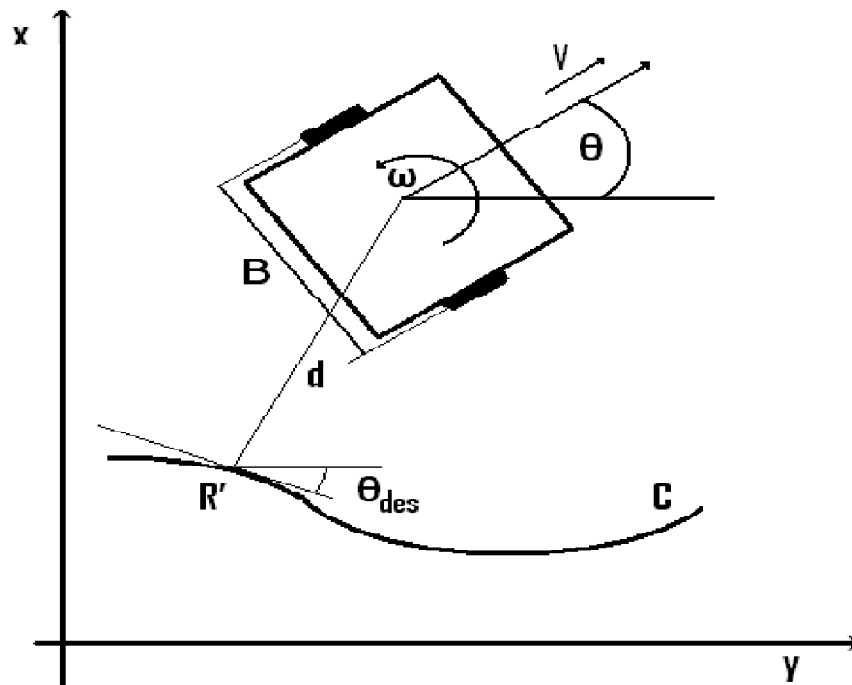


Figure 6: Path Followed by robot



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

$$\theta e = \theta - \theta_{des} \quad (23)$$

It is given in its simplest form by

$$\omega = v \cdot (p(sR') - k_1 \cdot d - k_2 \cdot \theta_e) \quad (24)$$

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 can be obtained in the case of a straight path

$$k_1 = \xi^2 \quad (25)$$

$$k_2 = 2\zeta\xi$$

### 5.3. The proposed approach of the Predictive Control

#### 5.3.1. 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].

$$J(k) = \sum_{n=0}^N [\hat{z}(k+n) - r(k+n)]^T Q [\hat{z}(k+n) - r(k+n)] + \lambda \cdot \omega^2(k+n) \quad (26)$$

#### 5.3.2. The control functions and the generated paths

To have good possibilities to avoid obstacles, we used linear type control functions  $u(v, x)$  in Figs. 7 (a)-(b). The parameters  $p_1$  and  $p_3$  denote the speed and turning respectively to achieve in and  $p_2$   $p_4$  the parameters of the speed and the turning to reach by. is the horizon of predictions.

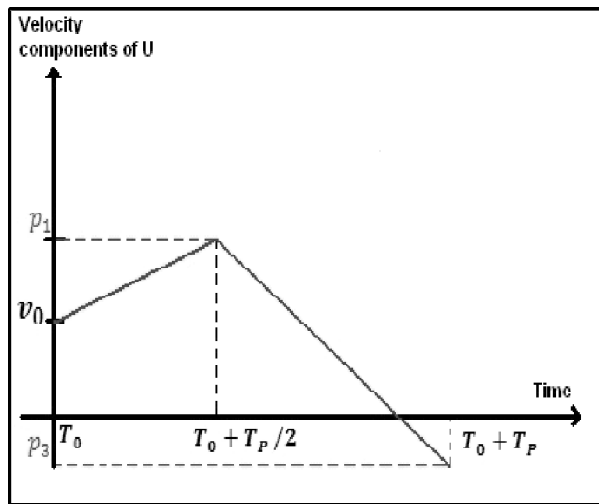


Figure 7 (a): Input functions for the speed

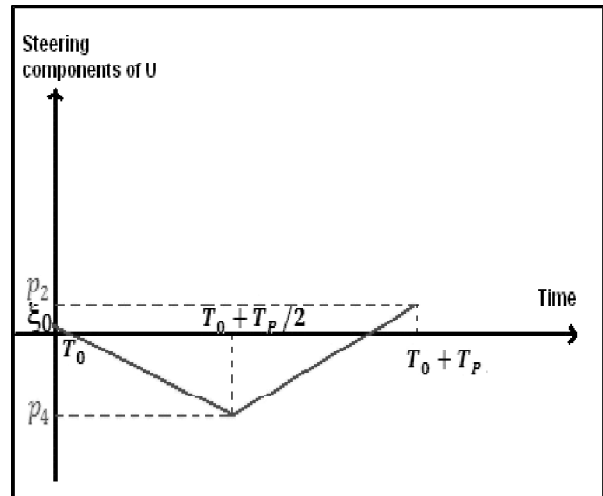


Figure 7 (b): Input functions for the steering

#### 5.3.2. Constraints

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

$$\begin{aligned}
 v_{\min} &\leq p_1 \leq v_{\max} \\
 v_{\min} &\leq p_3 \leq v_{\max} \\
 \xi_{\min} &\leq p_2 \leq \xi_{\max} \\
 \xi_{\min} &\leq p_4 \leq \xi_{\max}
 \end{aligned} \tag{27}$$

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

$$\begin{aligned}
 v_0 - decel_{\max} \times \frac{T_p}{2} &\leq p_1 \leq v_0 + accel_{\max} \times \frac{T_p}{2} \\
 p_1 - decel_{\max} \times \frac{T_p}{2} &\leq p_3 \leq p_1 + accel_{\max} \times \frac{T_p}{2} \\
 \xi_0 - \xi_{\max} \times \frac{T_p}{2} &\leq p_2 \leq \xi_0 + \xi_{\max} \times \frac{T_p}{2} \\
 p_2 - \xi_{\max} \times \frac{T_p}{2} &\leq p_4 \leq p_2 + \xi_{\max} \times \frac{T_p}{2}
 \end{aligned} \tag{28}$$

From this group of control functions, a group of trajectories is generated as shown in fig. 8. By simply varying each parameter values, we obtain a wide variety of movement possibilities.

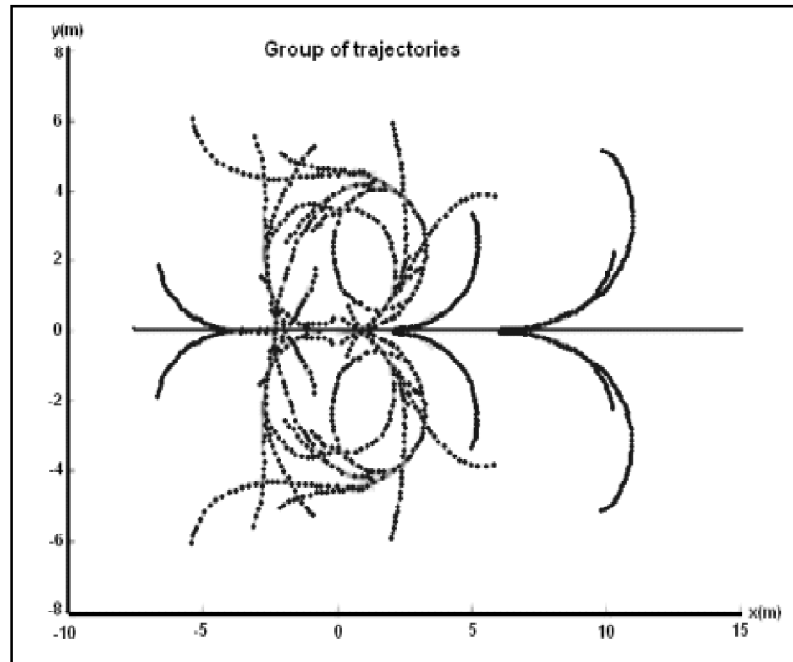


Figure 8: Setting the parameters of the predictive control

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:

$$\dot{s} = \vartheta \cdot \cos(\theta - \psi) \tag{29}$$

$$\dot{d} = \vartheta \cdot \sin(\theta - \psi) \tag{30}$$

$$\dot{\theta} = \omega \tag{31}$$

Where  $s$  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

$$\begin{aligned} \dot{d} &= v \cdot \theta_e \\ \dot{\theta}_e &= \omega \end{aligned} \quad (32)$$

And can be discretized with a sampling period

$$\begin{aligned} d(k+1) &= d(k) + h \cdot v \cdot \left[ \theta(k) - \psi(k) + \frac{h}{2} \right] \\ \theta(k+1) &= \theta(k) + h \cdot \omega(k) \end{aligned} \quad (33)$$

The equations can be writing in state spaces representation

$$\begin{pmatrix} d(k+1) \\ \theta(k+1) \end{pmatrix} = \begin{pmatrix} 1hv \\ 01 \end{pmatrix} \begin{pmatrix} d(k) \\ \theta(k) \end{pmatrix} + \begin{pmatrix} \frac{h^2}{2}v \\ h \end{pmatrix} \omega(k) + \begin{pmatrix} 0 & -hv \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ \psi(k) \end{pmatrix} \quad (34)$$

Or in compact form it is writing

$$z(k+1) = B \cdot z(k) + B_\omega \cdot \omega(k) + B_r \cdot r(k) \quad (35)$$

$$\begin{pmatrix} d(k) \\ \theta(k) \end{pmatrix} = z(k) : \text{System State} \quad (36)$$

$$\begin{pmatrix} 0 \\ \psi(k) \end{pmatrix} = r(k) : \text{The reference vector} \quad (37)$$

The criterion chosen to minimize the horizon is as following

$$J(k) = \lambda \cdot \omega^2(k+n) + \sum_{n=0}^N [\hat{z}(k+n) - r(k+n)]^T Q [\hat{z}(k+n) - r(k+n)] \quad (38)$$

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

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

$$J(k) = [\hat{Z}(k) - R(k)]^T I_Q [\hat{Z}(k) - R(k)] + \lambda \Omega^T(k) \Omega(k) \quad (39)$$

Where

$$\hat{Z}(k) = \begin{pmatrix} \hat{z}(k \setminus k) \\ \vdots \\ \hat{z}(k+N \setminus k) \end{pmatrix} = Fz(k) + G_\omega \cdot \Omega(k) + G_r r(k) \quad (40)$$

$$I_Q = \begin{pmatrix} Q & 0 & 0 & \cdots & 0 \\ 0 & Q & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & Q & 0 \\ 0 & \cdots & 0 & 0 & Q \end{pmatrix} \quad (41)$$

Where

$$\Omega(k) = (\omega(k) \cdots \omega(k+N))^T \quad (42)$$

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

$$= \begin{pmatrix} 0 \\ \Psi(k) \end{pmatrix}^T = \begin{pmatrix} 0 & \dots & 0 \\ \psi(k) & \dots & \psi(k+N) \end{pmatrix}^T \quad (43)$$

And

$$G_i = \begin{pmatrix} 0 & 0 & \dots & 0 & 0 \\ B_i & 0 & \dots & 0 & 0 \\ AB_i & B_i & \dots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & 0 \\ A^{N-1}B_i \dots & AB_i B_i & 0 & 0 & 0 \end{pmatrix} \quad (44)$$

$$F = (IA \dots A^N)^T \quad (45)$$

### 5.3.3. The predictive algorithm

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

$$\Omega(k) = -K_z \cdot z(k) - L_r \cdot R(k) \quad (46)$$

Where

$$L_z = (\lambda + G_\omega^T I_Q G_\omega)^{-1} G_\omega^T I_Q G_\omega F \quad (47)$$

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

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

$$\omega(k) = -k_\theta \cdot \theta(k) - k_d \cdot d(k) - K_\psi \cdot \Psi(k) \quad (48)$$

Where  $k_d$  and  $k_\theta$  are scalar and vector  $K_\psi$  as  $[1 \times (N + 1)] \psi$  multiplying the reference vector  $\psi(k)$ .

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

$$\psi(k) = ((\psi_1 \dots \psi_1, \psi_2 \dots \psi_2)^T \quad (49)$$

## 6. IMPLEMENTATION AND ROBOT CONSTRUCTION

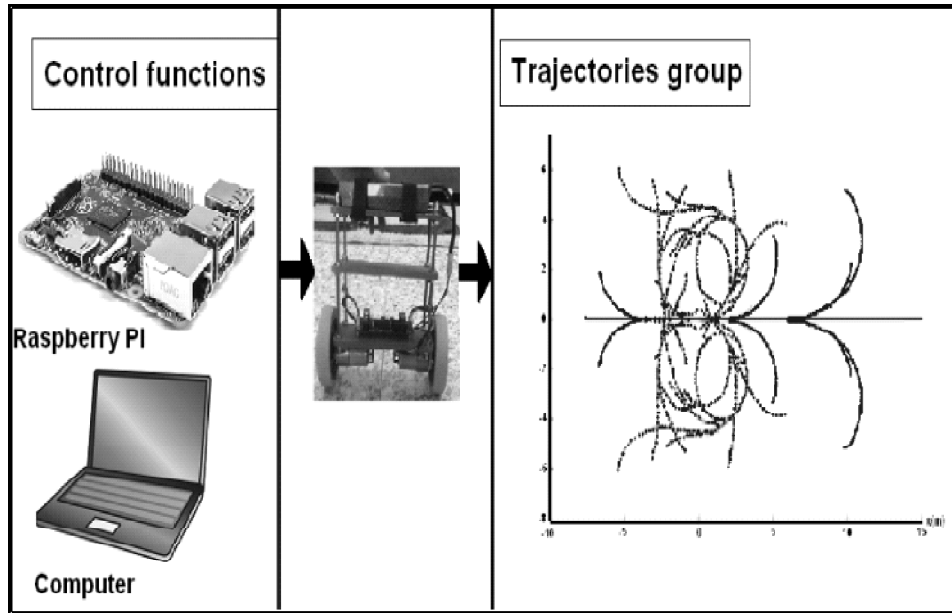
### 6.1. Architecture

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 Figure 9.

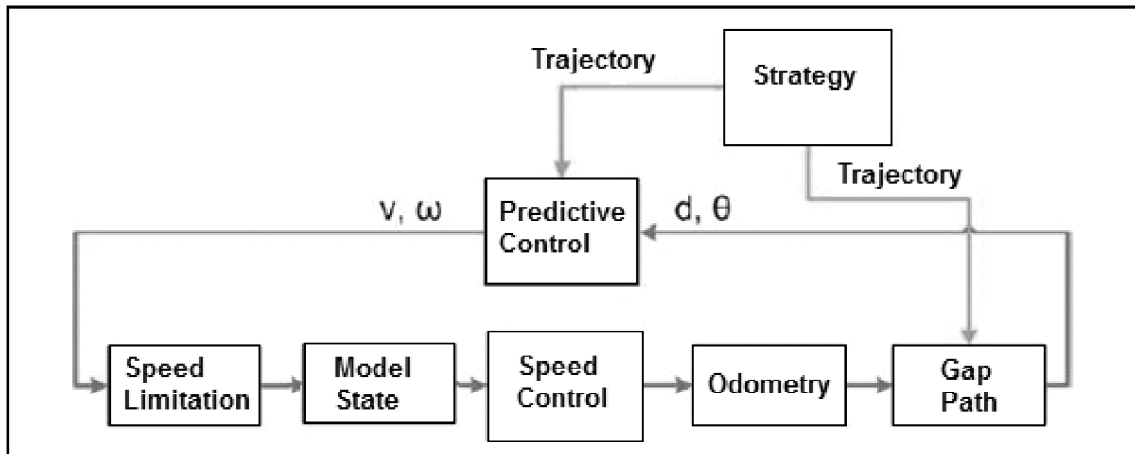
The synthesis of the two regulators in cascade was made and discussed. We must now put it in common and do all the required adaptations. The block diagram in Figure10 shows the general principle of the trajectory control.

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 software Matlab Simulink simulation. This optimizes some parameters to save time on the actual implementation.

The best results of simulations are shown in Figure 11. We see the reference path dotted 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 hunt although the way.



**Figure 9: System Architecture**



**Figure 10: General functional regulation**

## 6.2. 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.

## 6.3. Hardware

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.

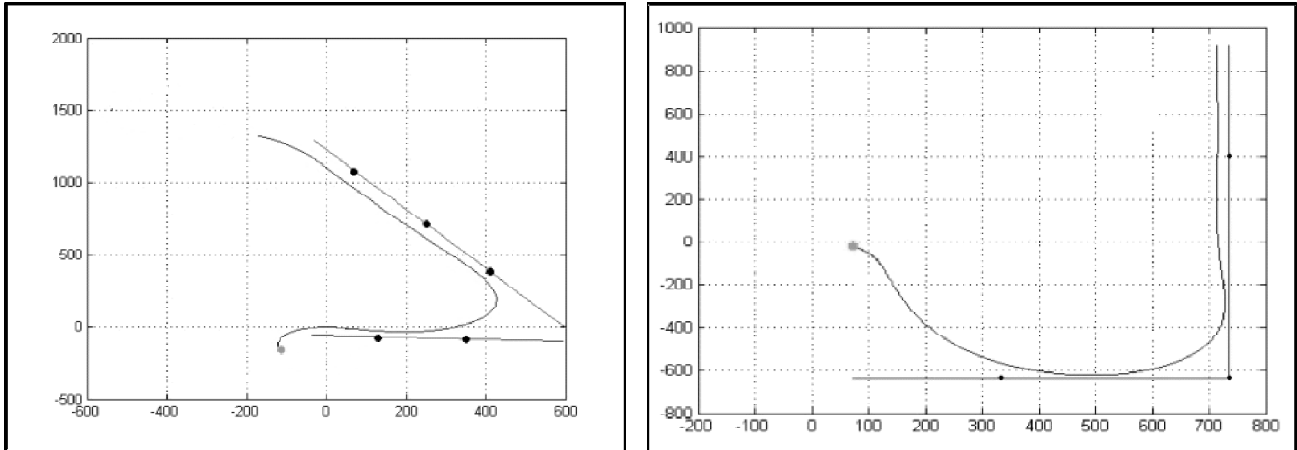


Figure 11: Simulation Results

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, SCL) but can also control another downstream device with AUX and AUX-DA-CL. Its consumption is low with 6 activated sensors.

The sensor has a 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 Figure 12.

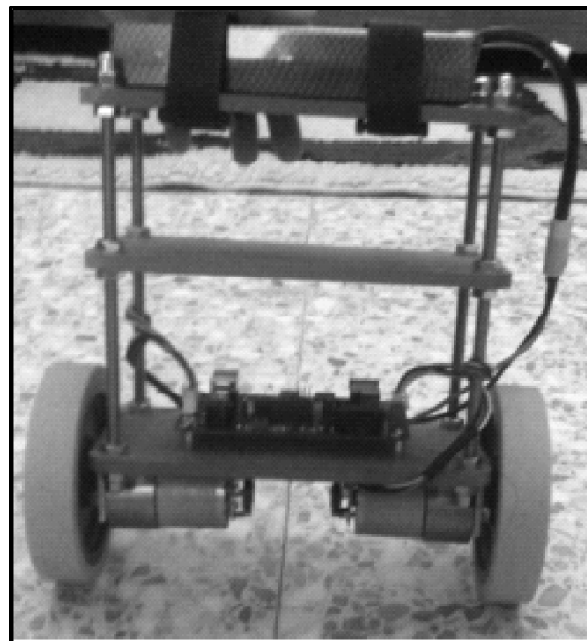


Figure12: Assembled parts of the Robot

#### 6.4. Software Implementation

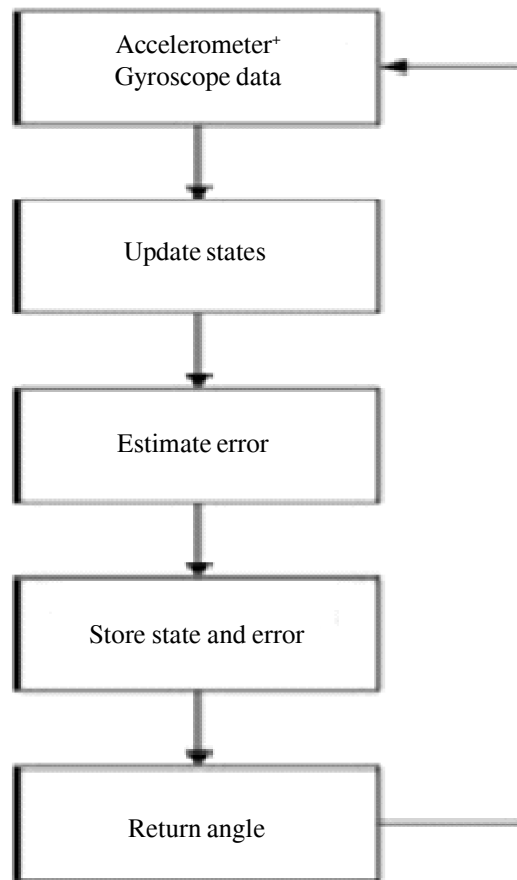
After having assembled the different hardware and sensors, we need to know how to exploit the different information. The gyroscope provides a measure of the angular velocity. By integrating this measure, we can get

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 Figure 13. 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.



**Figure 13: Kalman Process**

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 Figure 14.

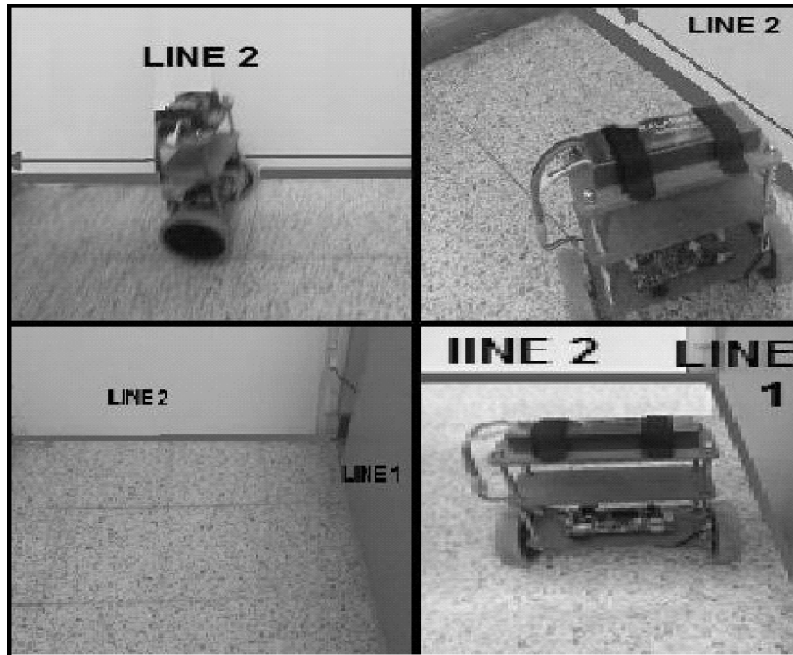


Figure14: The robot following the path and responding to constraints

## 7. CONCLUSION

The purpose of this work was to control a robot on a path. The robot was controlled 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 for implementation on the real system. We also presented the design of a vision control algorithm; our goal has been to integrate this sensor on a service robot performing movements in human internal environment. Thus two major steps in the design of the system were presented in this paper, the development and validation of the processing algorithm for the trajectory tracking and obstacles detection and secondly, the design and implementation of the architecture on a dedicated system. It was essential to choose a methodology that achieves the best compromise between performance and resources.

## REFERENCES

- [1] F. Grasser, S. Colombi and A. 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, The University of Western Australia School of Mechanical Engineering, Faculty of Engineering and Mathematical Sciences, University of Western Australia, 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), pp.141-154;2009.
- [5] Y. Takita, H. Date and 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," Student Conference on Research and Development, 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, 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, vol. 6764, 2008.
- [9] J.C. Latombe, Robot Motion Planning, Kluwer Academic Publishers, Norwell, MA, USA, 1991.
- [10] H. Choset, Sensor Based Motion Planning the Hierarchical Generalized Voronoi Graph, PhD thesis, California Institute of Technology, 1996.
- [11] B. Chazelle and L.J. Guibas, "Visibility and intersection problems in plane geometry," *Discrete and Computational Geometry*, 4 (6), 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, 32 (12), 1044-1054, 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, 2007.
- [14] A. Stenz, "Optimal and efficient path planning for partially-known environments," Proceedings of the International Conference on Robotics and Automation, 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 (9), 1094—1106, 2009.
- [16] I. Skrjanc and S. Blazic, "Predictive functional control based on fuzzy model : design and stability study", *Journal of Intelligent and Robotic Systems*, 43 (2), 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 and Robotic Systems*, 60 (2), 317-337, 2010.
- [18] A. Micaelli, "Trajectory tracking for unicycle-type and two-steering-wheels mobile robots," Sophia Anitpolis, France: Institut National de Recherche en Automatique er en Automatique, 1993.
- [19] J.S. Hu, M.C. Tsai, F.R. Hu and Y. Hori, "Robust control for coaxial two-wheeled electric vehicle," *Journal of Marine Science and Technology*, 18 (2), 172-180, 2010.
- [20] G. Chi, J. Hausbach and B. Hunter, Segbot: 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, Vol. 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] V.T. Pham, S. Jafari, C. Volos, A. Giakoumis, S. Vaidyanathan and T. Kapitaniak, "A chaotic system with equilibria located on the rounded square loop and its circuit implementation," IEEE Transactions on Circuits and Systems-II: Express Briefs, 63 (9), 2016.
- [27] S. Vaidyanathan and S. Sampath, "Anti-synchronisation of identical chaotic systems via novel sliding control and its application to a novel chaotic system," *International Journal of Modelling, Identification and Control*, 27 (1), 3-13, 2017.
- [28] S. Vaidyanathan, K. Madhavan and B.A. Idowu, "Backstepping control design for the adaptive stabilization and synchronization of the Pandey jerk chaotic system with unknown parameters," *International Journal of Control Theory and Applications*, 9 (1), 299-319, 2016.

- [29] R.K. Goyal, S. Kaushal and S. Vaidyanathan, "Fuzzy AHP for control of data transmission by network selection in heterogeneous wireless networks," *International Journal of Control Theory and Applications*, 9 (1), 133-140, 2016.
- [30] C.K. Volos, D. Prousalis, I.M. Kyprianidis, I. Stouboulos, S. Vaidyanathan and V.T. Pham, "Synchronization and anti-synchronization of coupled Hindmarsh-Rose neuron models," *International Journal of Control Theory and Applications*, 9 (1), 101-114, 2016.