International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-5 Issue-2, May 2016

Design and Visio Control for Navigation and Obstacles Detection Based on Color and Texture Attributes of Two-Wheeled Mobile Robot

Sabri M. Ben Mansour, Jawhar Ghommam, Saber M. Naceur

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

I. INTRODUCTION

The problem of nonholonomic 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.

Revised Manuscript Received on 30 May 2016.
***** Correspondence Author
Sabri M. Ben Mansour, LTSIRS, Remote Sensing and Spatially Referenced Information Systems Laboratory, National Engineering School of Tunis, Tunisia.
Jawhar Ghommam, CEM-Laboratory, National Engineering School of Sfax, sfax, Tunisia.
Saber M. Naceur, LTSIRS, Remote Sensing and Spatially Referenced Information Systems Laboratory, National Engineering School of Tunis, Tunisia.

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 nonholonomic constraints and other control goals (obstacle avoidance, minimum travel time, and minimum fuel consumption) are implicitly included in the path-planning procedure [7,8].

The first path planning method consists in the robot environment decompose in cells [9] (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 [10] or the visibility graphs [11]. 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 [10].

Other planning algorithms work in a discrete environment. Among the best known A * [11] which is planning an optimal path knowing the environment or D * [12] which is planning a path dynamically. 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 [13] 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 [14]. Interesting work from [15] was about the design of an optimal trajectory wich is made based on Bernstein-bézier curve and optimization. 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.

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.

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 [16]. 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 kinematic model and the control design are calculated in section 3. The speed and trajectory control are developed in section 4. In Section 5, implementation and robot construction is discussed. The algorithm for obstacles detection is described in Section 6. The conclusions are stated in Section 7.

II. 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 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 non holonomic constraint of the form

$$\dot{\gamma} \cdot \cos\theta(t) - \dot{\varkappa} \cdot \sin\theta(t) = 0 \qquad (1)$$

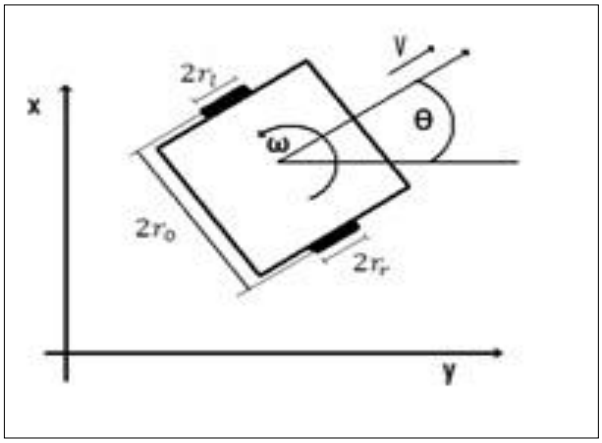

Figure1. Mobile inverted pendulum

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

Where $\dot{\boldsymbol{\theta}}_{\mathbf{r}}$ (t) and $\dot{\boldsymbol{\theta}}_{\mathbf{l}}$ (t) are the angular velocity of the right wheel and the angular velocity of the left wheel.

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

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

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

where

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

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

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

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 feas ible path by the robot, which does not make it collide with o bstacles, and to enable it to better follow the path provided b y the planner. It means making a selection in a path that the robot is able to perform, and therefore determining a criterio n to make that choice. Kinematic constraints such as saturat ions of actuators or stops joint must be stated clearly. This ty pe of constraint will play on input functions of our robot, on its u command. The set of trajectories that satisfy these const raints is called the set of eligible trajectories. The typical pat h to follow is a series of straight lines. The robot has to antic ipate the next turn with a radius of curvature, to be rotatable about an obstacle.

III. THE KINEMATIC MODEL AND THE CONTROL DESIGN

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

3.1 The robot 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 identica l parameters.

3.1.1 Mechanical equation of the motor

Consider the general characteristics of the engine we find:

$$U_i(t) = R_i \cdot i_i(t) \qquad (9)$$

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

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

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

$$M_i^1(t) = k_i \cdot n_i \cdot i_i(t) \quad (12)$$

$$M_i^2(t) = -F_i(t) \cdot r_i \quad (13)$$

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

3.1.2 Motion equations of the robot

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

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

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

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

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

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

$$\dot{X} = A \cdot X + B \cdot U$$
$$\dot{Y} = C \cdot X + D \cdot U \quad (19)$$

Then we get the explicit equations

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

$$\begin{pmatrix} \theta_r \\ \theta_l \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \theta_r \\ \dot{\theta}_r \\ \theta_l \\ \dot{\theta}_l \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} u_r \\ u_l \end{pmatrix} \quad (21)$$

where

θ_r : The angular position of the right wheel

θ_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 (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.

$$X(k+1) = \Phi \cdot X(k) + \Gamma \cdot U(k)$$
$$Y(k) = C \cdot X(k) + D \cdot U(k) \quad (22)$$

Φ , Γ, C, D: matrices giving by Matlab calculation

3.2 Control design

In the literature, the control law for the tracking path is base d on the assumption that the robot speed limits are directly tr ansmitted [17,18]. This hypothesis corresponds to a control i n perfect speed.

This separation between low-level (robot speed control) an d 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.

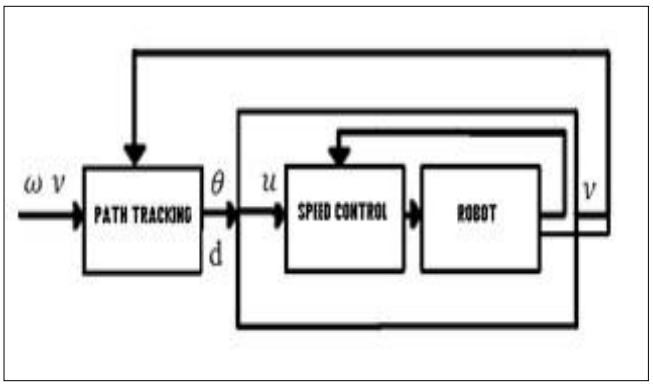

Figure 2. The two controllers in cascade

IV. 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 [19] that responds to the needs perfectly as shown in bloc diagr am in Figure 3.

The cost function to be minimized is:

$$J = \frac{1}{2}\sum_{k=0}^{N} [x^T(k)Q_1x(k) + u^T(k)Q_2u(k)] \quad (23)$$

We must therefore choose the matrices Q1 and Q2. They are selected positive diagonals. The weight of the position is ze ro. The weight of the speeds is much higher than the voltage of the engine.

$$Q_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 100'000 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100'000 \end{pmatrix} \quad (24)$$

$$Q_2 = \begin{pmatrix} 10 & 0 \\ 0 & 10 \end{pmatrix} \quad (25)$$

The dlqr control Matlab gives the optimum gain. Thus, the v oltage applied to closed loop is:

$$U(k) = K_G . X(k) \quad (26)$$

Where KG is a Gain obtained by Matlab calculation

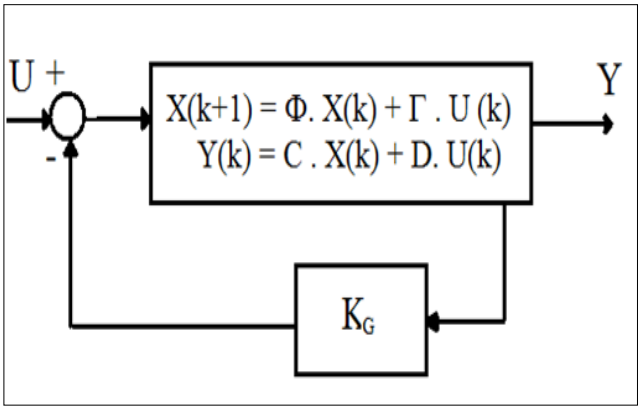

Figure 3. Block diagram of the control speed

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

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, 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 Figure 4.

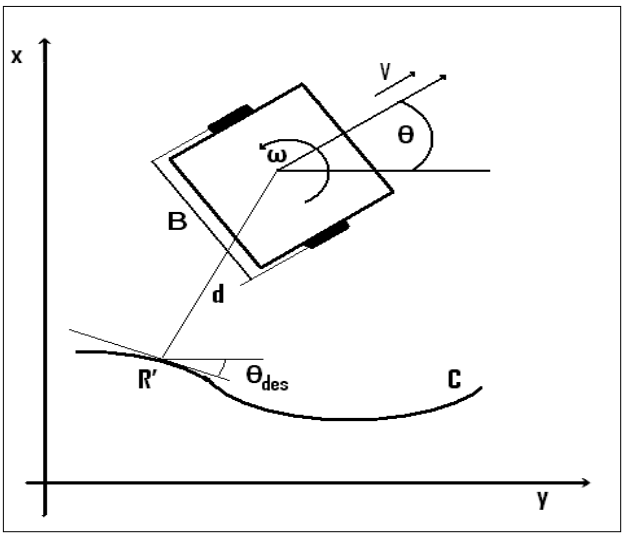

Figure 4. Path Followed by robot

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

$$\theta_e = \theta - \theta_{des} \quad (27)$$

It is given in its simplest form by:

$$\omega = \upsilon \cdot (p(sR') - k_1 \cdot d - k_2 \cdot \theta_e) \quad (28)$$

Where sR' is the curvilinear abscissa of the point R_0. 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:

$$\begin{aligned} k_1 &= \xi^2 \\ k_2 &= 2\zeta\xi \end{aligned} \quad (29)$$

4.3 Our 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 [20] [21] [22].

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

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, □) in Figures 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 $T_0 + T_P$. TP is the horizon of predictions.

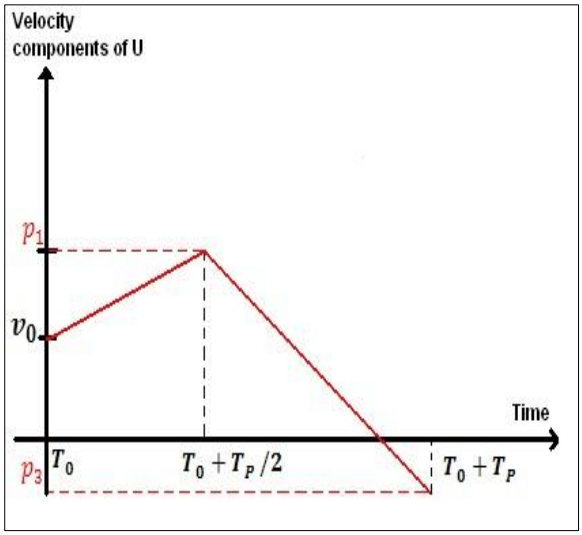

Figure 5 (a). Input functions for the speed

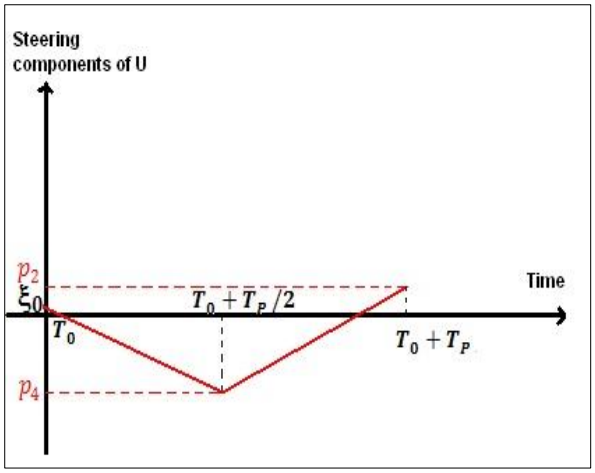

Figure 5 (b). Input functions for the steering

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}):

$$\begin{aligned} v_{min} \le p_1 \le v_{max} \\ v_{min} \le p_3 \le v_{max} \\ \xi_{min} \le p_2 \le \xi_{max} \\ \xi_{min} \le p_4 \le \xi_{max} \end{aligned} \quad (31)$$

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

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

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

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

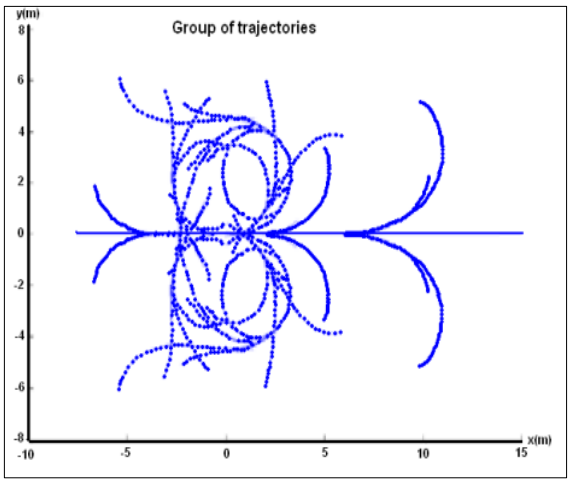

Figure 6. Setting the parameters of the predictive control

The Predictive control requires a linear kinematic model giv ing in [23]. It is based on the following model, consisting for monitoring a straight line:

$$\dot{s} = \vartheta \cdot \cos(\theta - \psi) \quad (33)$$
$$\dot{d} = \vartheta \cdot \sin(\theta - \psi) \quad (34)$$
$$\dot{\theta} = \omega \quad (35)$$

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:

$$\dot{d} = \upsilon \cdot \theta_e$$
$$\dot{\theta}_e = \omega \quad (36)$$

And can be discretized with a sampling period h:

$$d(k+1) = d(k) + h \cdot \upsilon \cdot \left[\theta(k) - \psi(k) + \frac{h}{2}\right]$$
$$\theta(k+1) = \theta(k) + h \cdot \omega(k) \quad (37)$$

The equations can be writing in state spaces representation:

$$\begin{pmatrix} d(k+1) \\ \theta(k+1) \end{pmatrix} = \begin{pmatrix} 1 & h\upsilon \\ 0 & 1 \end{pmatrix} \begin{pmatrix} d(k) \\ \theta(k) \end{pmatrix} + \begin{pmatrix} \frac{h^2}{2}\upsilon \\ h \end{pmatrix} \omega(k) + \begin{pmatrix} 0 & -h\upsilon \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ \psi(k) \end{pmatrix} \quad (38)$$

Or in compact form it is writing:

$$z(k+1) = B \cdot z(k) + B_\omega \cdot \omega(k) + B_r \cdot r(k) \quad (39)$$

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

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

The criterion chosen to minimize the horizon is as followin g:

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

Where $\hat{z}$ is the predicted output, Q is the Weighting matrix, λ is a scalar weight and N is the horizon. λ : A scalar weight Using (39) in (42), we have the following:

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

Where

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

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

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

$$R(k) = \left(r(k) \cdots r(k+N)\right)^T = \begin{pmatrix} 0 \\ \Psi(k) \end{pmatrix}^T = \begin{pmatrix} 0 & \cdots 0 \\ \psi(k) \cdots & \psi(k+N) \end{pmatrix}^T \quad (47)$$

And

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

$$F = (I\ A\ \ldots.\ A^N)^T \quad (49)$$

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).

$$\Omega(k) = -L_z z(k) - L_r R(k) \quad (50)$$

Where

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

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:

$$\omega(k) = -k_\theta\theta(k) - k_d d(k) - K_\Psi\Psi(k) \quad (52)$$

Where k_d and k_θ are scalar and vector K_ψ as $[1 \times (N+1)]$ ψ multiplying the reference vector ψ(k).In the case of a turn (sequence of two lines), the vector reference has the following form:

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

$$\psi(k) = ((\psi_1 \cdots \psi_1, \psi_2 \cdots \psi_2)^T \quad (53)$$

V. IMPLEMENTATION AND ROBOT CONSTRUCTION

5.1 Architecture

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

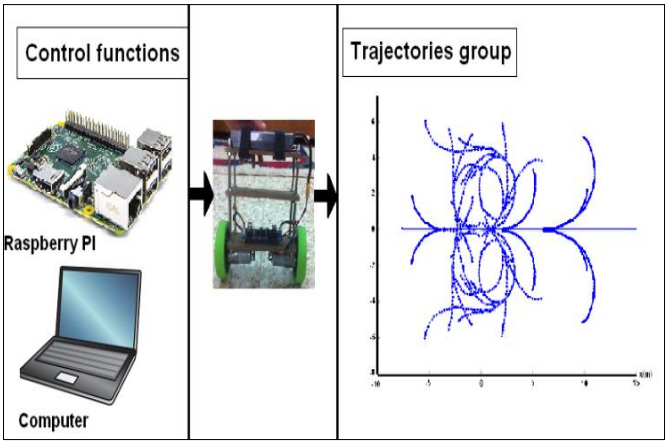

Figure 7. System Architecture

The synthesis of the two regulators in cascade was made and discussed. We must now put it in common and do all the re quired adaptations. The block diagram in Figure 8 shows the general principle of the trajectory control.

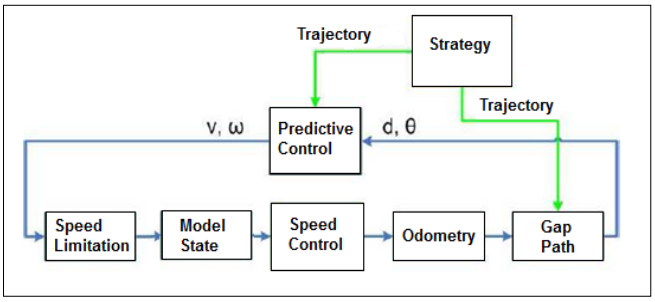

Figure 8 General functional regulation

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

The best results of simulations are shown in Figure 9. We se e the reference path dotted line and how far the robot follow s the line. It is noted that even with an initial position outsid e path (the angle is also set to a divergent path), the robot hu nt although the way.

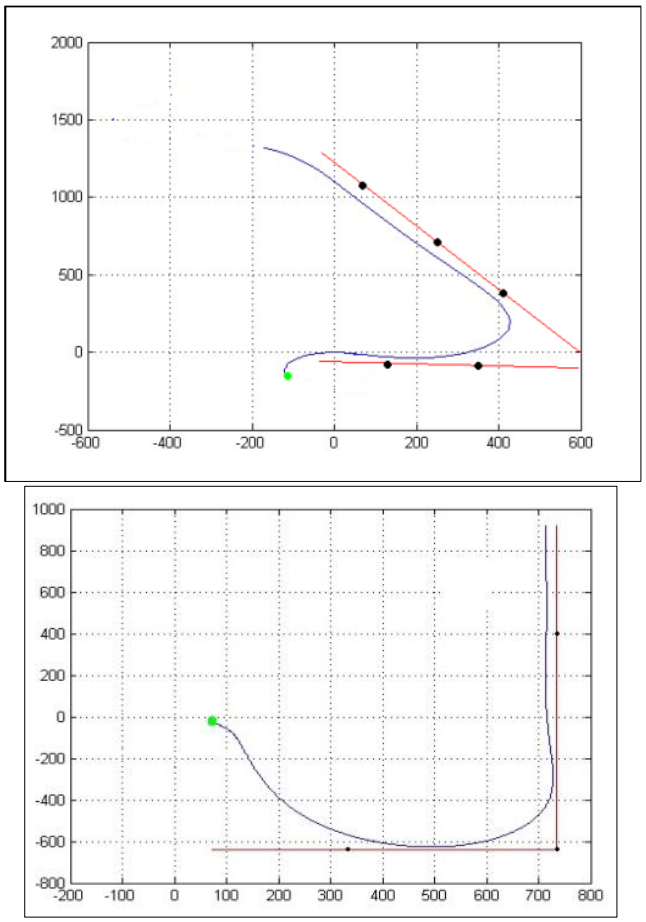

Figure 9. Simulation Results

Figure 10 shows a simulation for robot navigation moving s urrounded 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 accou nt the size of the robot. The robot path is represented by the blue line, each small cross marking a new iteration of the na vigation algorithm (period Te).

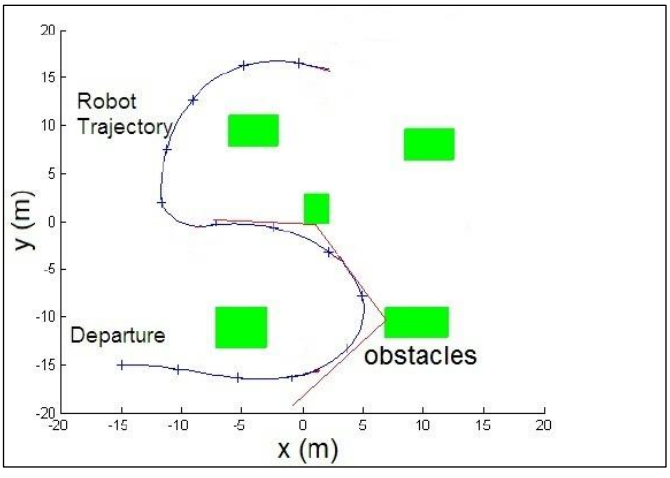

Figure 10. Simulation Results for obstacles avoidance

The following simulation shows the behavior of the robot in the case where the trajectory of Reference cross an obstacle.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

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 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 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 10.

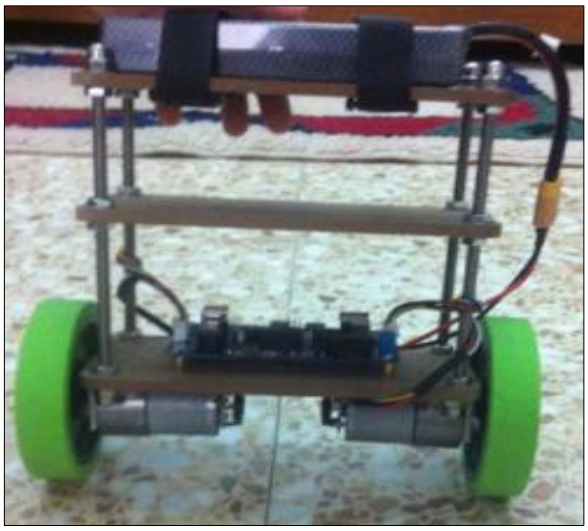

Figure 11. Assembled parts of the Robot

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 Figure 11. 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.

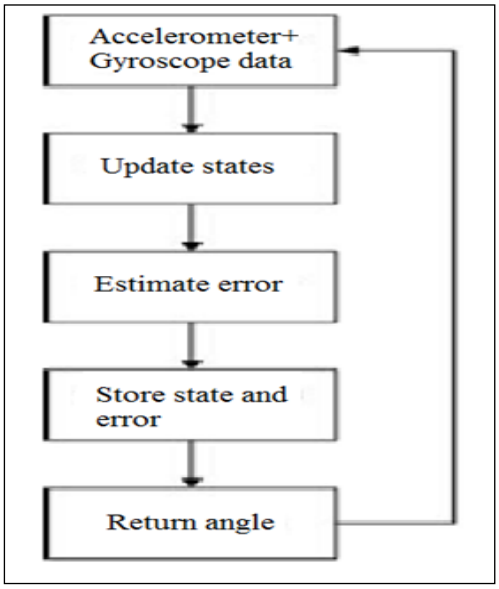

Figure 12. Kalman Process

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 12.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

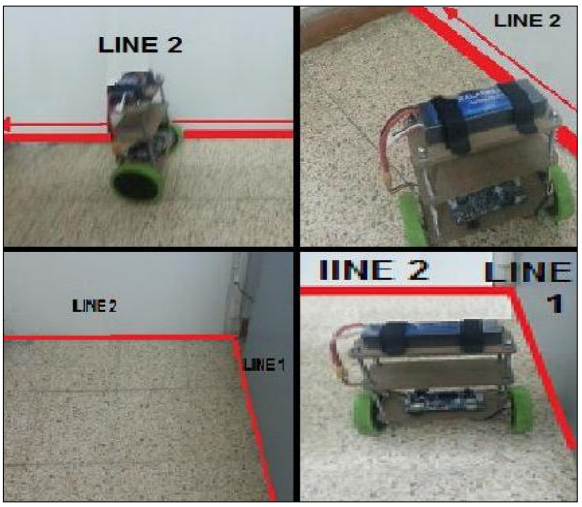

Figure 13. The robot following the path and responding to constraints

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.

VI. ALGORITHM FOR OBSTACLES DETECTIONS

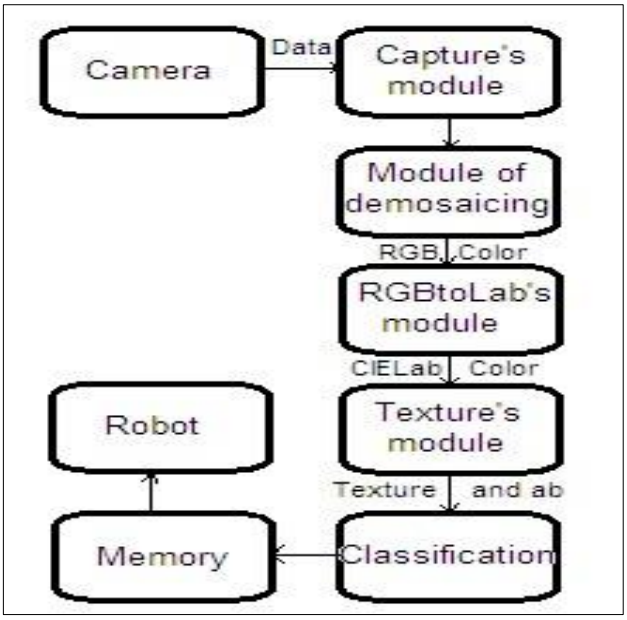

Figure 14. Algorithm diagram for the obstacles detection

Figure 13 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.

6.1 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:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} . \begin{pmatrix} R \\ G \\ B \end{pmatrix} \quad (54)$$

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

$$L^{*}=116\left[f\left(\frac{Y}{Y_w}\right)\right]\text{-}16 \quad (55)$$

$$a=500\left[f\left(\frac{X}{X_w}-f\left(\frac{Y}{Y_w}\right)\right)\right] \quad (56)$$

$$b=200\left[f\left(\frac{Y}{Y_w}-f\left(\frac{Z}{Z_w}\right)\right)\right] \quad (57)$$

where Xw, Yw and Zw 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 Lo w-intensity space; it is defined as follows:

$$f\ (t)=\begin{cases} t^{\frac{1}{3}} & si\ t>\left(\frac{6}{29}\right)^3 \\ \frac{1}{3}\left(\frac{29}{6}\right)^2 t+\frac{4}{29} & si\ t<\left(\frac{6}{29}\right)^3 \end{cases} \quad (58)$$

6.2 Architecture for texture analysis

The architecture for the texture analysis consist of two main processes: sum and difference of images, and calculating tex ture attributes. These processes are shown inside two rectan gles in the Fig. 14. The rectangle at the top represents the m odulus of the sum and difference of histograms, the bottom r epresents the textural attributes itself divided into six modul es. 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 exec uted during the analysis of dense texture. Addition and subtr action are the operations performed between the current pixe l and the displaced pixels. Sixteen movements in total are us ed for four distances and four orientations, such as (1, 2, 3 a nd 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 : $\frac{1}{2}\sum_i i.\ \widehat{P_s}($ i), variance: $\frac{1}{2}\left(\sum_i(i-2\mu)^2.\ \widehat{P_s}(i)+\sum_j j^2.\ \widehat{P_D}(j)\right)$, uniformity: $\sum_j \frac{1}{1+J}.\ \widehat{P_D}(j)$, contrast : $\sum_j j^2.\ \widehat{P_D}(j)$ and correlation: $\frac{1}{2}($ $\sum_i(i-2\mu)^2.\ \widehat{P_s}\ (i)\text{-}\sum_j j^2.\ \widehat{P_D}(j)).$

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

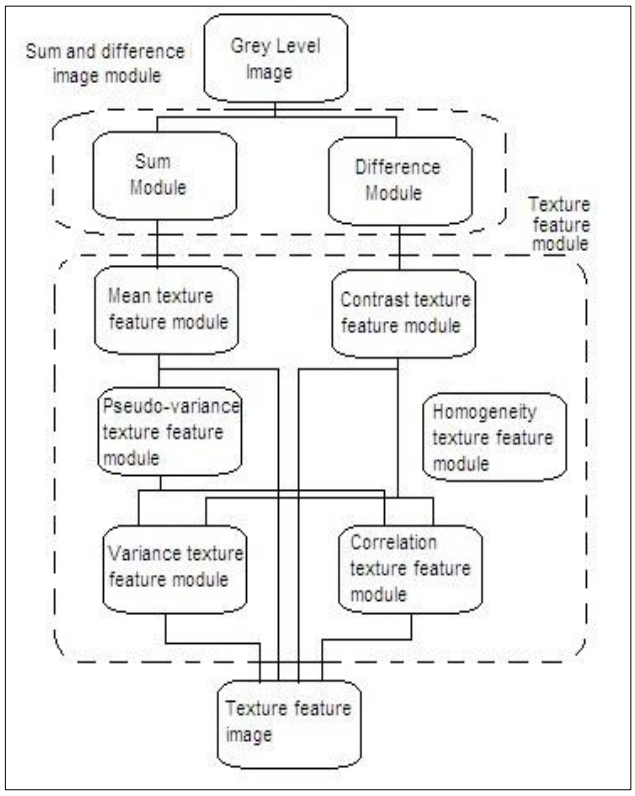

Figure 15. Architecture diagram for texture analysis based on sums and differences of histograms

6.3 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 t wo because it is the detection of a particular object in the sce ne and not the identification of each of the objects. The dat abase was created from sample images of some obstacles an d some parts of the ground. One file contains the samples in the space of the color attributes and texture calculated from t he base image. Each sample is marked with a label indicatin g whether it belongs to the class of interest (soil) or not (obst acles).

6.4 Experimental results

The images were acquired with tested digital camera STMic roelectronics. Obstacle detection was performed by using th e results of evaluations in the previous section. The classific ation method used is the discrete AdaBoost algorithm with 2 0 weak assumptions. Modeling soil was made using 22 attri butes of color and texture. The final method for the detectio n of obstacles was implemented on an embedded architectur e. Thus, we chose to make dense obstacle detection (soil in b lue and obstacles in red), so the treatment is done pixel by pi xel. Results are in Figure 15 (a) and (b).

Figure 16. (a) Original images

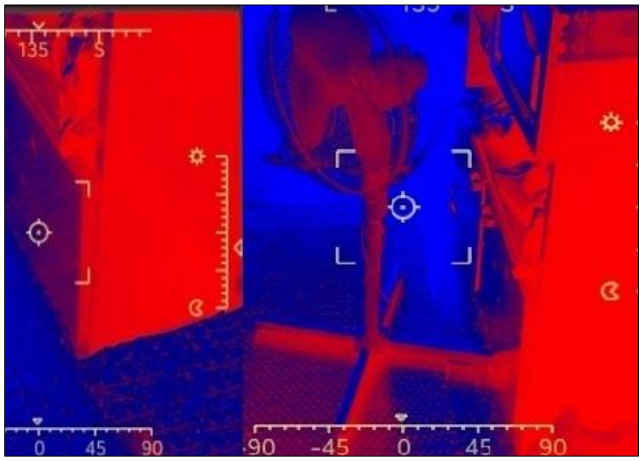

Figure 16. (b) Images resulting from the classification

VII. CONCLUSION

The purpose of this work was to control a robot on a path. T he 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 allo wed to find the correct settings required for implementation on the real system. We also presented the design of a visio c ontrol algorithm; our goal has been to integrate this sensor o n a service robot performing movements in human internal e nvironment. Thus two major steps in the design of the syste m were presented in this paper, the development and validati on of the processing algorithm for the trajectory tracking an d obstacles detection and secondly, the design and implemen tation of the architecture on a dedicated system. It was essen tial to choose a methodology that achieves the best compro mise between performance and resources. The simulations a nd implementation validate the effectiveness of our predictiv e control in crowded map. The trajectories calculated from t he direct kinematic model of the robot and respecting its ow n 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 c ontraints and to the path provided by the planner.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

ACKNOWLEDGMENTS

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

VIII. APPENDIX

8.1 Appendix 1

$F_i(t)$	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	Reduction factor
$M_i^3(t)$	Torque caused by the wheel's dynamic
m_o	Mass of the robot
R_i	Motor resistance
r_i	Radius of the wheel
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_o	Angular position of the robot

8.2 Appendix 2

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

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

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

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

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

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

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

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

REFERENCES

1. Grasser F, D'arrigo A, Colombi S and Rufer A. Joe: A Mobile Inverted Pendulum:IEEE Transaction on Industrial Electronics;2002.
2. Ooi R C. 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. Ho K C R. Balancing Wheeled Robot: Research Project, University of Southern Queensland, Australia; 2005.
4. Grepl R. Balancing Wheeled Robot: Effective Modelling, Sensory Processing And Simplified Control: Engineering Mechanics, 16 (2), pp.141–154;2009.
5. Takita Y, Date H and Shimazu H. 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. Chee Y O and Abidin M S Z. Design and Development of Two Wheeled Autonomous Balancing Robot; Student Conference on Research and Development; 2006.
7. Nawawi S W, Ahmad M and Osman J H S. Real-Time Control of a Two-Wheeled Inverted Pendulum Mobile Robot: World Academy of Science, Engineering and Technology,214-220; 2008.
8. Tsai C, Chan C and Fan Y H. Planned Navigation of a Self-balancing Autonomous Service Robot: IEEE International Conference on Advanced Robotics and Its Social Impacts, vol 6764; 2008.
9. Latombe J.C Kluwer,Robot Motion Planning: Academic Publishers, Boston;1991.
10. Choset. Sensor Based Motion Planning the Hierarchical Generalized Voronoi Graph" Phd thesis, California Institure of Technology;1996.
11. Chazzzelle and Guibas.Visibiliy and ontersection problems in plane geometry", Discrete and Computational Geometry;2000
12. Shin and McKay. A dynamic programming approach to trajectory planning of the robotic manipulators", IEEE Transactions on Automatic Control;1996.
13. Hart, A Formal Basis for the Heuristic Determination of Minimum Cost Paths", IEEE Transactions on Systems Science and Cybernetics;1986.
14. Stentz and Anthony, Optimal and Efficient Path Planning for Partially-Known Environments, Proceedings of the International Conference on Robotics and Automation;1994.
15. Defoort, Motion planning for cooperative unicycle-type mobile robots with limited sensing ranges, A distributed receding horizon approach , Robotics and Autonomous Systems;2009.
16. ŠKRJANC, Igor, BLAŽIČ, Sašo, Predictive functional control based on fuzzy model : design and stability study", Journal of intelligent & robotic systems, ISSN 0921-0296, 2005, vol. 43, pp. 283-299 ;2005.
17. KLANČAR, Gregor, ŠKRJANC, Igor, 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, ISSN 0921-0296, Nov. 2010, vol. 60, no. 2, pp. 317-337, ilustr., doi: 10.1007/s10846-010-9417-8;2010.
18. Micaelli A. 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. Hu J S, Tsai M C, Hu F R and Hori Y. Robust Control For Coaxıal Two-Wheeled Electrıc Vehıcle: Journal of Marine Science and Technology, pp 172-180;2010.
20. Chi G, Hausbach J and Hunter B. Segbot: Senior Design Project, University of Illinois at Urbana-Champaign, USA ; 2005.
21. Bock H.G and Plitt K. A multiple shooting algorithm for direct solution of optimal control problems. 9th IFAC world congress Budapest. Pergamon Press; 1984.
22. Klancar G, Skrjanc I. Tracking-error model-based predictive control for mobile robots in real time: Robotics and Autonomous Systems; 2007.
23. Courtial E. Commande predictive et estimation d'état de systèmes non linéaires : Rapport de thèse université Claude Bernard – Lyon ;1996.
24. Vallius T and Röning J. Embedded Object Concept: Case Balancing Two-Wheeled Robot: Proceedings of the SPIE, Vol. 6764; 2007.
25. Bak M. Poulsen K, Ravn O. Path Following Mobile Robot in the Presence of Velocity Constraints: Kongens Lyngby, Denmark : Technical University of Denmark; 2000

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)