RACBOT-RT: ROBUST DIGITAL CONTROL FOR DIFFERENTIAL SOCCER-PLAYER ROBOTS

João Monteiro and Rui Rocha
ISR – Institute of Systems and Robotics, Department of Electrical and Computer Engineering
University of Coimbra, 3030-290 Coimbra, Portugal

Keywords: Digital control, mobile robots, non-holonomic, Lyapunov stability convergence, robot soccer.

Abstract: In the field of robot soccer, mobile robots must exhibit high responsiveness to motion commands and possess precise pose control. This article presents a digital controller for pose stability convergence, developed to small-sized soccer robots. Special emphasis has been put on the design of a generic controller, which is suitable for any mobile robot with differential kinematics. The proposed approach incorporates adaptive control to deal with modeling errors and a Kalman filter which fuses odometry and vision to obtain an accurate pose estimation. Experimental results are shown to validate the quality of the proposed controller.

1 INTRODUCTION

This paper describes the work that is being done in the field of digital control and real time systems for mobile robots within the RACbot-RT M.Sc. project. Many approaches for differential control of mobile robots have been presented. For instance, (A. Gholipour, 2000) presents a generic controller where pose estimation is extracted from the robot's kinematics, and an adaptive control block is introduced to deal with modelling errors. In (Y. Kanayama, 1990), a Lyapunov based nonlinear kinematic controller is presented where the influence of the control parameters is studied, without giving emphasis to modeling errors. The present approach brings together the simplicity of the Lyapunov mathematical laws, the adaptive control concept to deal with modeling errors and proper fusion of two sensorial data – vision and odometry – for robust pose estimation (T. Larsen, 2000).

2 ROBOT DYNAMIC MODEL

Based on the Lagrange's mathematical modelation of mechanical systems, on (A. Gholipour, 2000), and considering $G(q) = C(q, \ddot{q}) = 0$, the dynamic equations of the mobile robot can be written as

$$\begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I \end{bmatrix} = \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{h} \end{bmatrix} = \tag{1}$$

$$\frac{1}{R}\begin{bmatrix} \cos(h) & \cos(h) \\ \sin(h) & \sin(h) \\ L & -L \end{bmatrix} \cdot \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} + \begin{bmatrix} sin(h) \\ cos(h) \\ 0 \end{bmatrix} \lambda, \tag{2}$$

where τ_1 and τ_2 are the left and right motor torques respectively, m and I are the robot's mass and inertia, R is the wheel's radius and L is the line distance between the two wheels. The non-holonomic restriction is deduced from (1) and given by the equation

$$\dot{x}\sin(h) - \dot{x}\cos(h) = 0, \tag{3}$$

from where it is imposed that a non-holonomic mobile robot can only move in the direction normal to the axis of the driving wheels.

3 TRAJECTORY DEFINITION AND ROBOT KINEMATICS

Our 2D path planner defines a trajectory as a time variant pose vector represented in the playing field, which has its own global cartesian system defined. The robot in the world possesses three degrees of freedom, which are represented by the actual pose vector

$$q = \begin{bmatrix} x \\ y \\ h \end{bmatrix}, \tag{4}$$

where x, y are the robot's coordinates and h is its heading. The latter is defined positively in the counter-clockwise direction, beginning at the positive xx axis. The state q_0 is denoted as the zero pose state $(0,0,2n\pi)$, where n is an integer value. Since the robot is capable of moving in the world, the pose q is a function of time t. The movement of the robot is controlled by its linear and angular velocities, v and ω respectively, which are also functions dependent of t. The robot's kinematics is defined by the following Jacobian matrix

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{h} \end{bmatrix} = \dot{q} = Jp = \begin{bmatrix} cos(h) & 0 \\ sin(h) & 0 \\ 0 & 1 \end{bmatrix} q, \tag{5}$$

where the velocity matrix is defined by

$$p = \begin{bmatrix} v \\ w \end{bmatrix} \tag{6}$$

This kinematics is common for all non-holonomic robots.

4 POSE ERROR

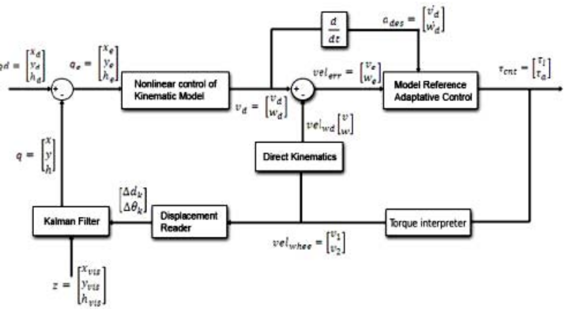
To implement the controller, two pose vectors need to be defined: the actual pose of the robot already represented in (4), and the desired pose vector represented by

$$q_d = \begin{bmatrix} x_d \\ y_d \\ h_d \end{bmatrix}, \tag{7}$$

which, by definition, is the target pose for the robot to achieve at the end of its movement. We will define the pose error q_e as the transformation of the reference pose q_d to the local coordinate system of the robot with origin (x_c, y_c), where the actual robot's absciss is given by h_c's amplitude. Such transformation is the difference between q_d and q_c,

$$q_e = \begin{bmatrix} \dot{x_e} \\ \dot{y_e} \\ \dot{h_e} \end{bmatrix} = \begin{bmatrix} cos(h_c) & sin(h_c) & 0 \\ -sin(h_c) & cos(h_c) & 1 \\ 0 & 0 & 1 \end{bmatrix} .(q_d - q_c) \tag{8}$$

One can easily see that if $q_d = q_c$, the pose error is null, being this the ideal final state.



$qd = \begin{bmatrix} x_d \\ y_d \\ h_d \end{bmatrix}$
$q_e = \begin{bmatrix} x_e \\ y_e \\ h_e \end{bmatrix}$
Nonlinear control of Kinematic Model
$v_d = \begin{bmatrix} v_d \\ w_d \end{bmatrix}$
$\frac{d}{dt}$
$a_{des} = \begin{bmatrix} v'_d \\ w'_d \end{bmatrix}$
$vel_{err} = \begin{bmatrix} v_e \\ w_e \end{bmatrix}$
Model Reference Adaptative Control
$\tau_{cnt} = \begin{bmatrix} \tau_l \\ \tau_a \end{bmatrix}$
$vel_{wd} \begin{bmatrix} v \\ w \end{bmatrix}$
Direct Kinematics
$q = \begin{bmatrix} x \\ y \\ h \end{bmatrix}$
$\begin{bmatrix} \Delta d_k \\ \Delta \theta_k \end{bmatrix}$
Kalman Filter
Displacement Reader
Torque interpreter
$vel_{whee} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$
$z = \begin{bmatrix} x_{vis} \\ y_{vis} \\ h_{vis} \end{bmatrix}$

Figure 1: Control scheme.

5 DIGITAL CONTROLLER DESIGN

The controller is designed in three parts. In the first, kinematic stabilization is achieved using nonlinear control laws. For the second, the acceleration is used for exponential stabilization of linear and angular velocities. The uncertainties related with the robot's physical structure modeled parameters are compensated using adaptive control. For the final part, pose estimation is made fusing odometry and vision by means of a Kalman filter. The latter was designed in a way that independence of the mathematical system's model is achieved.

The developed approach is depicted in Fig. 1. It is a feedback controller, in which the input state is the desired robot's pose $[x_d\ y_d\ h_d]'$. At its output, proper update of the torques for each wheel is done to fulfill the controller's objective. Next, we will explain the relevant blocks.

5.1 Pose Error Generator

The error dynamics is written independently of the inertial (fixed) coordinate frame by Kanayama transformation. Expanding (8), we have

$$q_e = \begin{bmatrix} x_e \\ y_e \\ h_e \end{bmatrix} = \begin{bmatrix} \cos(h) & \sin(h) & 0 \\ -\sin(h) & \cos(h) & 1 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x_d - x \\ y_d - y \\ h_e - h \end{bmatrix} \tag{9}$$

which will compose the pose error vector.

5.2 Nonlinear Kinematic Controller

Lyapunov based nonlinear controllers are very simple and yet, at the same time, very successful in kinematic stabilization. So, bringing together the concepts simplicity and functionality, the Lyapunov stability theorem proved to be of great utility for this project. Based on such theorem, the deduced equations for the desired linear and angular velocities of the platform are,

$$v_d = v_r \cos(h_e) - K_x x_e \tag{10}$$

$$w_d = w_r + K_y v_r y_e + K_h \sin(h_e), \tag{11}$$

where K_x, K_y and K_h are positive constants. By La Salle's principle of convergence and proposition 1 of (Y. Kanayama, 1990), the null pose state q_0 is always an equilibrium state if the reference velocity is higher than zero ($v_r > 0$). This way, we can have three weighting constants for the pose error, without interfering in the overall pose stability of the robot.

5.3 Model Reference Adaptive Control

The motivation to include this block comes from the need to alter the control laws used by the controller for it to cooperate with parameter uncertainties. Based on (A. Gholipour, 2000), one can extract the adaptation rules for the linear velocity,

$$\frac{d\theta_1}{dt} = -\varepsilon_1 e \dot{v}_d \Leftrightarrow \theta_1 = \int -\varepsilon_1 e \dot{v}_d dt \tag{12}$$

$$\frac{d\theta_2}{dt} = -\varepsilon_2 e \dot{v}_d \Leftrightarrow \theta_2 = \int -\varepsilon_2 e \dot{v}_d dt. \tag{13}$$

where v_d is the desired linear velocity, and e the velocity error. The parameters ε_1 and ε_2 are manually tuned for best performance achievement.

Identically, similar rules for the angular velocity can be found.

5.4 Kalman Filter

A differential robot with odometry system as in our case, is equipped with an encoder in each motor. An angular displacement of α radians on the rotor corresponds to a performed distance d on the periphery of the wheel, and subsequently to an encoder count. The distance is given by $d = k\alpha$ with $k = \frac{1}{r}$, where r is the wheel radius. If the robot's movement is assumed to be linear, the distances d_1 and d_2 performed by the left and right wheels respectively, can be transformed in linear and angular displacements. For a particular sample instant, we have:

$$\Delta d_k = \frac{d_{1,k} + d_{2,k}}{2} \quad ; \quad \Delta h_k = \frac{d_{1,k} - d_{2,k}}{b}. \tag{14}$$

The robot's coordinates referenced on the world's coordinates can be determined by the following equations:

$$X_{k+1} = X_k + \Delta d_k \cos(h_k + \frac{\Delta h}{2}) \tag{15}$$

$$Y_{k+1} = Y_k + \Delta d_k \sin(h_k + \frac{\Delta h}{2}) \tag{16}$$

$$h_{k+1} = h_k + \Delta_k. \tag{17}$$

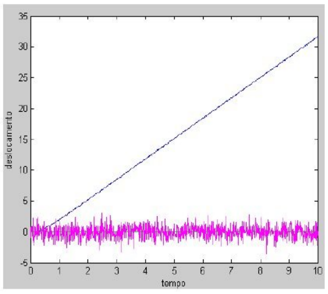


Figure 2: Filter test case results.

These coordinates constitute the state vector, and are observed by the vision coordinate vector z. These measurements can be described as a nonlinear function c of the robot's coordinates, which possesses an independent noise vector v. Defining the above equations as vector α and placing Δd_k and Δh_k in an input vector u_k, the robot can be modeled by the following equations

$$x_{k+1} = a(x_k, u_k, w_k, k) \tag{18}$$

$$z_k = c(x_k, v_k, k), \tag{19}$$

where $w_k\tilde{}N(0, Q_k)$ and $v_k\tilde{}N(0, r_k)$, being both not correlated, i.e., $E[w_l v_l^T] = 0$.
We can now design the extended Kalman filter, using the odometry-based system model:

$$\hat{x}_{k+1} = a(x_k, u_k, w_k, k) \tag{20}$$

$$P_{k+1} = A_k P_k A_k^T + Q_k \tag{21}$$

$$K_k = P_k C_k^T [C_k P_k C_k^T + R_k]^{-1} \tag{22}$$

$$\hat{x}_k = \hat{x}_k + K_k [z_k - C_k \hat{x}_k] P_k = [I - K_k C_k] P_k \tag{23}$$

The process noise is modeled by two Gaussian white noises applied on the two odometry displacement measurements Δd_k and Δh_k.
– Filter simulation test case: Robot in $x = 0$, $y = 0$, heading $= 0$, $\sigma^2_{vis} = 1$, $\sigma^2_{odo} = 1$
In this simulation, the displacement made by the robot in open loop will be indefinitely linear along the xx axis. Fig. 2 shows this situation, being the blue slope the displacement over xx, the red slope the displacement over yy and the green one, the robot's heading. The magenta slope represents the vision error. As we can see, the filter possesses little but visible sensitivity to vision noise.

6 ON-THE-FIELD RESULTS

– Setpoint (final target position) command to (0,0)
For this test, we send a *setPosition* command to $(0,0)$. In Fig. 3(A), we see a screenshot of the visualizer tool developed for the controller module. The robot

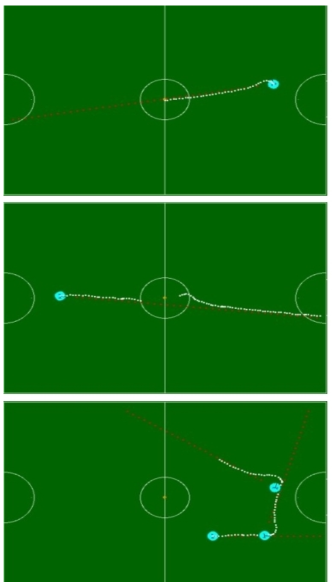


Figure 3: (A) – *SetPoint* command to $x = 0, y = 0$. (B) – *setVelocity* with $v = 0.3mps$ and desired *velocity angle* $= 0$. (C) – Sequence of *setVelocity* commands with $v = 0.3mps$ and *velocity angle* $= 1.3rad$.

accurately goes to the defined *setPoint*, but possessing a *yy* axis precision error of 1 to 2 centimeters maximum. This precision error exists because of two main causes. The first is the backward force exerted by the energy cable that feeds the robot in test environment[1]. The second is because of the defined tuned parameter for the influence of the robot's error over the *yy* coordinate – K_y. Tuning for near-zero error is possible but leads to a very *hard* control scheme in the presence of a physical disturbance, making the controller produce high overshoot for compensation. Since our robot is to walk on a field where collisions with other robots may be present, the revealed accuracy perfectly suits for our needs. For collision-free applications, where minimum physical errors exist and depending on the world's space, we can make the control *harder*, raising K_y.

– setVelocity (velocity vector) with v = 0.3mps and desired velocity angle = 0
For this test, the robot was subjected to extreme noise conditions.

Referring to Fig. 3(B), the robot is subjected to two disturbances done by blocking its left wheel, evident by the multiple white dots in the same place. Also, we blocked the color code of the robot used to be identified by the vision module for a while, so that no vision data was being received by the controller for it to estimate the actual pose. Controller's robustness is proved.

– Sequence of setVelocity commands with $v = 0.3mps$ *and velocity angle* $= 1.3rad$
In this final test (Fig. 3(C)), we sent a sequence of *setVelocity* commands to evaluate the control module's response in the presence of new velocity instructions. This test approaches from our real application target, the soccer game, where a high movement dynamic is required for the robot. Referring to Fig. 3(C), a velocity vector with zero desired angle is first sent, followed by two *setVelocitie*'s with the same module and 1.3 *rad* for the desired *velocity angle*. Finally, the robot is halted with a *halt* instruction. As we can conclude, the robot accurately executes the performed commands, evidencing the software module's robustness.

[1]During tests, we prefer to have the robot constantly fed with energy instead of placing the batteries that discharge with time.

7 CONCLUSIONS

A controller for pose error elimination of a soccer-player robot was projected and its practical results have been shown. For the theoretical basis of the controller, particular interest was given to create a general scheme independent of the mathematical extraction of the test structure. On-the-field tests reveal that the projected approach is not only valid, but also robust. Despite the fact that the Real Time System has not already been implemented, the idea of placing it in the system is planned and will be present in the final version of this project.

REFERENCES

A. Gholipour, M. J. Y. (2000). Dynamic tracking control of nonholonomic mobile robot with model reference adaptation for uncertain parameters. University of Tehran.

T. Larsen, M. B. (2000). Location estimation for an autonomously guided vehicle using an augmented kalman filter to autocalibrate the odometry. Technical University of Denmark.

Y. Kanayama, Y. Kimura, F. M. T. N. (1990). A stable tracking control method for an autonomous mobile robot. IEEE.