MODEL PREDICTIVE CONTROLLER FOR TRAJETORY TRACKING BY DIFFERENTIAL DRIVE ROBOT WITH ACTUATION CONSTRAINTS

Marcos R. O. A. Maximo*

*Instituto Tecnológico de Aeronáutica
Praça Marechal Eduardo Gomes, 50, Vila das Acácias, 12228-900
São José dos Campos, SP, Brasil

Email: maximo.marcos@gmail.com

Abstract— This work addresses the problem of trajectory tracking by a differential drive mobile robot subjected to actuation constraints. The robot's dynamical system is nonlinear, but linearization around a reference trajectory yields a linear time-varying system. To actively deal with the actuation constraints and the time-varying dynamics, Model Predictive Control (MPC) is employed to design the controller. Finally, we show simulation results to validate and analyze the proposed method.

Keywords— Model Predictive Control, Robot Control, Differential Drive Mobile Robot.

1 Introduction

Differential drive mobile robots are very popular due to their simplicity (Siegwart et al., 2011). These robots have only two actuated wheels, but other unactuated wheels are added to ensure stability. A differential drive robot is able to turn by commanding different speeds for each powered wheel.

The Very Small Size (VSS) competition is a robotic soccer challenge where two teams with three autonomous robots each compete (7th Latin American IEEE Student Robotics Competition, 2008). VSS imposes strict dimensions limits: every robot must fit inside a cube of side 7.5 cm. Given this constraint, most teams uses differential drive robots. A vision system composed of an overhead camera and computer vision algorithms is able to track colored marks on top of each robot, measuring its position and orientation. Furthermore, most of the processing occurs in a central computer that communicates wirelessly with the robots.

One important ability of a VSS robot is to quickly move throughout the field while avoiding other players. This paper documents the development of a trajectory tracking controller for ITAndroids' VSS team. ITAndroids is the robotics competition group from Instituto Tecnológico de Aeronáutica (ITA). The group also has experience with other robotics competitions, namely RoboCup's Soccer 2D, Soccer 3D and Humanoid Kid-Size leagues and IEEE Standard Educational Kit (SEK) category.

A common approach in Mobile Robotics is to separate the locomotion decision process in two layers: planning and control. First, the robot plans a trajectory to get to its destination, then a controller reasons about the actual physical commands needed to follow the planned trajectory. The planning problem is usually solved by employing a search algorithm, such as A* or Rapidly Exploring Random Tree (RRT) (Siegwart et al., 2011). On the other hand, the trajectory tracking controller is typically designed using techniques from Control Theory (Kanjanawaniskul, 2012; Klančar and Škrjanc, 2007; Kühne et al., 2005; Camacho and Alba, 2007). In this work, our interest rests in designing a controller for following a trajectory, thus we assume a previously planned trajectory.

Model Predictive Control (MPC) is an optimal control approach where the future behavior of the system predicted by a model is taken into account. MPC is mostly a discrete time technique and a optimization problem is solved at each time step by considering a fixed number of future steps (finite horizon). State and actuation constraints may be easily incorporated into the MPC formulation by adding constraints to the optimization variables. Nevertheless, constrained optimization problems are generally not analytically solvable, thus they require iterative algorithms, which makes MPC computationally intensive.

Nonlinear underactuated dynamics and actuation constraints make the trajectory tracking problem particularly challenging for classic control techniques. However, MPC looks promising for this problem, due to its capabilities to explicitely consider the future reference and actuation constraints (Kanjanawaniskul, 2012). Therefore, we decided to use MPC to design a trajectory tracking controller for ITAndroids' VSS team.

2 Robot Modeling

Our robot modeling is inspired by the Very Small Size Soccer (VSS) environment, especially by our own VSS differential drive robots. First, let the robot state (position and orientation) $\mathbf{q} = [x, y, \psi]^T$ be defined with respect to a global coordinate system. Moreover, we consider that each wheel is speed controlled, which in our case is

achieved by an embedded PID controller using encoder feedback.

Instantaneously, a differential drive robot in motion with linear velocity v and angular velocity ω is making a curve of radius R, as shown in Figure 1. Then, if the robot is not slipping:

$$\omega_L = \frac{v_L}{r} = \frac{\omega}{r}\left(R - \frac{l}{2}\right) = \left(v - \frac{\omega l}{2}\right)\frac{1}{r} \quad (1)$$

$$\omega_R = \frac{v_R}{r} = \frac{\omega}{r}\left(R + \frac{l}{2}\right) = \left(v + \frac{\omega l}{2}\right)\frac{1}{r} \quad (2)$$

where ω_L and ω_R are the angular velocities of the left and right wheels, respectively. Analogously, v_L and v_R relate to the linear velocities of the wheels. Moreover, the parameters l and r are the distance between the wheels and the radius of each wheel, respectively. Solving (1) and (2) for v and ω:

$$v = \frac{v_R + v_L}{2} = \left(\frac{\omega_R + \omega_L}{2}\right) r \quad (3)$$

$$\omega = \frac{v_R - v_L}{l} = \left(\frac{\omega_R - \omega_L}{l}\right) r \quad (4)$$

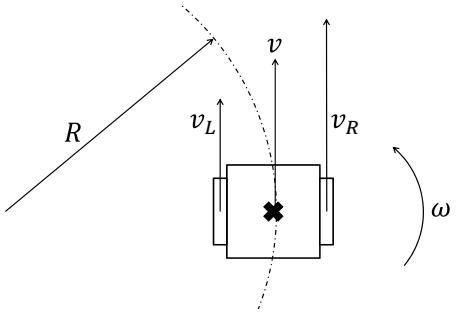

Figure 1: Differential drive mobile robot making a curve of radius R.

Therefore, we may transform between $[v, \omega]^T$ and $[\omega_L, \omega_R]^T$ using these equations. For convenience, controllers are usually designed considering $\mathbf{u} = [v, \omega]^T$ as input. Then, the motion kinematics of a differential drive robot yield the following nonlinear dynamical system:

$$\dot{\mathbf{q}}(t) = \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\psi}(t) \end{bmatrix} = \begin{bmatrix} v(t)\cos\psi(t) \\ v(t)\sin\psi(t) \\ \omega(t) \end{bmatrix} \quad (5)$$

where $\mathbf{q} = [x, y, \psi]^T$ is the robot kinematic state. In the Literature, this model is known as unicycle (Kanjanawaniskul, 2012). Note that the dynamical system presented in Equation (5) implicitly assumes that the wheel's speed controller has bandwidth high enough to let us neglect the wheel dynamics. Furthermore, the vision system directly measures the robot's kinematic state $\mathbf{q}$.

The robot's wheels are usually driven by DC motors, which have speed limits due to the battery voltage limit. Thus, we also consider:

$$-\omega_{MAX} \leq \omega_L \leq \omega_{MAX} \quad (6)$$

$$-\omega_{MAX} \leq \omega_R \leq \omega_{MAX} \quad (7)$$

3 Trajectory Tracking Problem

The trajectory tracking problem may be formally defined as making the robot follows a reference trajectory $\mathbf{x}_r(t) = [x_r(t), y_r(t)]^T$ defined in a time interval $[t_0, t_f]$ (Klančar and Škrjanc, 2007). For a given time instant $t \in [t_0, t_f]$, if we consider no state error and no perturbations, we may compute the required orientation $\psi_r(t)$ and the command $\mathbf{u}_r(t) = [v_r(t), \omega_r(t)]^T$ to maintain perfect trajectory tracking:

$$v_r(t) = (-1)^b \sqrt{\dot{x}_r^2(t) + \dot{y}_r^2(t)} \quad (8)$$

$$\psi_r(t) = \text{atan2}(\dot{y}_r(t), \dot{x}_r(t)) + b\pi \quad (9)$$

$$\omega_r(t) = \dot{\psi}_r(t) = \frac{\dot{x}_r(t)\ddot{y}_r(t) - \dot{y}_r(t)\ddot{x}_r(t)}{\dot{x}_r^2(t) + \dot{y}_r^2(t)} \quad (10)$$

where b defines the motion direction: $b = 0$ makes the robot moves forwards, while $b = 1$ makes it moves backwards. We may also define the reference state $\mathbf{q}_r(t) = [x_r(t), y_r(t), \psi_r(t)]^T$. Note that Equations (8), (9) and (10) require functions $x(t)$ and $y(t)$ to be twice differentiable and $v_r(t) \neq 0, \forall t \in [t_0, t_f]$. Moreover, the function $\text{atan2}(x, y)$ is the arctangent of x/y extended to the whole unit circle range.

4 Linearization Around a Reference Trajectory

Despite some minor notation differences, the derivation ahead closely follows the one shown in (Klančar and Škrjanc, 2007). Consider a virtual robot tha perfectly tracks the reference trajectory. This robot executes $\mathbf{u}_r(t)$. The real robot may have a state tracking error $\mathbf{e}(t) = [e_1(t), e_2(t), e_3(t)]^T$, which may be written in the real robot coordinates system as:

$$\mathbf{e} = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} (\mathbf{q}_r - \mathbf{q}) \quad (11)$$

Deriving Equation (11) and using the relationships provided by Equation (5):

$$\dot{\mathbf{e}} = \begin{bmatrix} \cos e_3 & 0 \\ \sin e_3 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_r \\ \omega_r \end{bmatrix} + \begin{bmatrix} -1 & e_2 \\ 0 & -e_1 \\ 0 & -1 \end{bmatrix} \mathbf{u} \quad (12)$$

We may think of the command $\mathbf{u}$ as composed by feedforward $\mathbf{u}_F$ and feedback $\mathbf{u}_B$ components, i.e. $\mathbf{u} = \mathbf{u}_F + \mathbf{u}_B$. The feedforward component keeps the robot following the trajectory assuming zero tracking error, thus intuitively we expect $\mathbf{u}_F$ to be equals to $\mathbf{u}_r$. However, taking into account the orientation error, we use instead:

$$\mathbf{u}_F = \begin{bmatrix} v_r \cos e_3 \\ \omega_r \end{bmatrix} \quad (13)$$

Substituting $\mathbf{u} = \mathbf{u}_F + \mathbf{u}_B$ in Equation (12)

$$\dot{\mathbf{e}} = \begin{bmatrix} \omega_r e_2 \\ -\omega_r e_1 + \sin e_3 v_r \\ 0 \end{bmatrix} + \begin{bmatrix} -1 & e_2 \\ 0 & -e_1 \\ 0 & -1 \end{bmatrix} \mathbf{u}_B \quad (14)$$

The dynamical system in Equation (14) is nonlinear, but note that the reference trajectory point, i.e. $\mathbf{e} = \mathbf{0}_{3\times1}$ and $\mathbf{u}_B = \mathbf{0}_{2\times1}$, is an equilibrium point, since:

$$\dot{\mathbf{e}}(\mathbf{e} = \mathbf{0}_{3\times1}, \mathbf{u}_B = \mathbf{0}_{2\times1}) = \mathbf{0}_{3\times1} \quad (15)$$

Therefore, we may linearize around the reference trajectory, which yields:

$$\dot{\mathbf{e}} = \begin{bmatrix} 0 & \omega_r & 0 \\ -\omega_r & 0 & v_r \\ 0 & 0 & 0 \end{bmatrix} \mathbf{e} + \begin{bmatrix} -1 & 0 \\ 0 & 0 \\ 0 & -1 \end{bmatrix} \mathbf{u}_B \quad (16)$$

Note that this dynamical system is linear time-varying, since $v_r(t)$ and $\omega_r(t)$ is time-varying.

5 Model Predictive Controller

Model Predictive Control (MPC) techniques solve an optimization problem at each time considering predicted future system behavior to yield an optimal control sequence (Camacho and Alba, 2007). Our MPC formulation has the following characteristics: linear time-varying model, discrete time implementation, quadratic cost function, use of receding horizon, and constraints handling.

Our MPC approach shares similarities with ones from other works (Klančar and Škrjanc, 2007; Kanjanawaniskul, 2012; Kühne et al., 2005), but we correctly deal with the physical constraints imposed by the wheels' speed limits. We use the linearized time-varying dynamics shown in Equation (16) to predict future system behavior.

5.1 Time Discretization

The linear time-varying dynamical system shown in Equation (16) may be discretized by using Euler approximation:

$$\dot{\mathbf{e}}(t) = \frac{\mathbf{e}(t+T) - \mathbf{e}(t)}{T} \quad (17)$$

To obtain the following discrete time system:

$$\mathbf{e}(k+1) = \mathbf{A}(k)\mathbf{e}(k) + \mathbf{B}(k)\mathbf{u}_B(k) \quad (18)$$

where:

$$\mathbf{A}(k) = \begin{bmatrix} 1 & \omega_r(k)T & 0 \\ -\omega_r(k)T & 1 & v_r(k)T \\ 0 & 0 & 1 \end{bmatrix} \quad (19)$$

$$\mathbf{B}(k) = \mathbf{B} = \begin{bmatrix} -T & 0 \\ 0 & 0 \\ 0 & -T \end{bmatrix} \quad (20)$$

5.2 Cost Function

We used a quadratic cost function (Klančar and Škrjanc, 2007; Camacho and Alba, 2007):

$$\begin{aligned} J = &\sum_{i=1}^{N} (\mathbf{e}_r - \hat{\mathbf{e}}(k+i|k))^T \mathbf{Q} (\mathbf{e}_r - \hat{\mathbf{e}}(k+i|k)) \\ &+ \sum_{i=1}^{M} \hat{\mathbf{u}}_B^T(k+i-1|k) \mathbf{R} \hat{\mathbf{u}}_B(k+i-1|k) \end{aligned} \quad (21)$$

where N is the length of the prediction horizon and M is the length of the control horizon, with $M \leq N$. $\mathbf{Q}$ e $\mathbf{R}$ are weight matrices for the penalizing the state error and the control effort, respectively. $\mathbf{Q} \in \mathbb{R}^3 \times \mathbb{R}^3$ and $\mathbf{R} \in \mathbb{R}^2 \times \mathbb{R}^2$, with $\mathbf{Q} \geq 0$ and $\mathbf{R} > 0$. Finally, $\mathbf{e}_r$ is the controller reference. In practice, we always use $\mathbf{e}_r = \mathbf{0}_{3\times1}$.

5.3 Prediction

Aplying Equation (18) recursively, we can predict future state errors:

$$\begin{aligned} \hat{\mathbf{e}}(k+i|k) = &\mathbf{\Lambda}(k, 0, i-1)\mathbf{e}(k) + \\ &\sum_{h=1}^{i} [\mathbf{L}(k, h, i)\hat{\mathbf{u}}_B(k+h-1|k)] \end{aligned} \quad (22)$$

where:

$$\mathbf{\Lambda}(k, \alpha, \beta) = \begin{cases} \prod_{j=\alpha}^{\beta} \mathbf{A}(k+j), \alpha \leq \beta \\ \mathbf{I}_{3\times3}, \alpha > \beta \end{cases} \quad (23)$$

$$\mathbf{L}(k, h, i) = \mathbf{\Lambda}(k, h, i-1)\mathbf{B}(k+h-1) \quad (24)$$

For our system, after reaching the desired state $\mathbf{e} = \mathbf{0}_{3\times1}$, no command $\mathbf{u}_B$ is needed to stay in this state (assuming no perturbations). Therefore, we adopt:

$$\hat{\mathbf{u}}_B(k+i-1|k) = \mathbf{0}_{2\times1}, M < i \leq N \quad (25)$$

Defining matrices $\hat{\mathbf{E}}$ e $\hat{\mathbf{U}}_B$:

$$\hat{\mathbf{E}}(k) = \begin{bmatrix} \hat{\mathbf{e}}(k+1|k) \\ \hat{\mathbf{e}}(k+2|k) \\ \vdots \\ \hat{\mathbf{e}}(k+N|k) \end{bmatrix} \quad (26)$$

$$\hat{\mathbf{U}}_B(k) = \begin{bmatrix} \hat{\mathbf{u}}_B(k|k) \\ \hat{\mathbf{u}}_B(k+2|k) \\ \vdots \\ \hat{\mathbf{u}}_B(k+M-1|k) \end{bmatrix} \quad (27)$$

Then, we may write:

$$\hat{\mathbf{E}}(k) = \mathbf{\Phi}(k)\mathbf{e}(k) + \mathbf{G}(k)\hat{\mathbf{U}}_B(k) \quad (28)$$

where $\mathbf{\Phi}(k) \in \mathbb{R}^{3N} \times \mathbb{R}^3$ e $\mathbf{G}(k) \in \mathbb{R}^{3N} \times \mathbb{R}^{2M}$ are matrices built from Equation (22):

$$\mathbf{\Phi}(k) = \begin{bmatrix} \mathbf{\Lambda}(k,0,0) \\ \mathbf{\Lambda}(k,0,1) \\ \vdots \\ \mathbf{\Lambda}(k,0,N-1) \end{bmatrix} \tag{29}$$

$$\mathbf{G}(k) = \begin{bmatrix} \mathbf{L}(k,1,1) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \mathbf{L}(k,1,N) & \cdots & \mathbf{L}(k,M,N) \end{bmatrix} \tag{30}$$

Finally, Equation (20) shows that $\mathbf{B}(k) = \mathbf{B}$ is independent of the sampling time, so the equations above may be simplified.

5.4 Constraints Handling

Combining Equations (1), (2), (6) e (7), we have:

$$-\omega_{MAX} r \leq \left(v - \frac{\omega l}{2} \right) \leq \omega_{MAX} r \tag{31}$$

$$-\omega_{MAX} r \leq \left(v + \frac{\omega l}{2} \right) \leq \omega_{MAX} r \tag{32}$$

which may be rewritten as:

$$\mathbf{Su} \leq \mathbf{b} \tag{33}$$

where:

$$\mathbf{S} = \begin{bmatrix} 1 & \frac{l}{2} \\ -1 & -\frac{l}{2} \\ 1 & -\frac{l}{2} \\ -1 & \frac{l}{2} \end{bmatrix} \tag{34}$$

$$\mathbf{b} = \begin{bmatrix} \omega_{MAX} r \\ \omega_{MAX} r \\ \omega_{MAX} r \\ \omega_{MAX} r \end{bmatrix} \tag{35}$$

In fact, our controller decides $\mathbf{u}_B$, thus:

$$\mathbf{S}(\mathbf{u}_F + \mathbf{u}_B) \leq \mathbf{b} \Rightarrow \mathbf{Su}_B \leq \mathbf{b} - \mathbf{Su}_F \tag{36}$$

Therefore, the controller should satisfy the following constraints:

$$\mathbf{S}_B \hat{\mathbf{u}}_B(k+i|k) \leq \mathbf{b}_B(k+i), 0 < i \leq M-1 \tag{37}$$

where:

$$\mathbf{S}_B = \mathbf{S} \tag{38}$$

$$\mathbf{b}_B(k) = \mathbf{b} - \mathbf{Su}_F(k) \tag{39}$$

5.5 Computing the Optimal Control

Let us define:

$$\mathbf{E}_r = [\mathbf{e}_r]_N \in \mathbb{R}^{3N} \tag{40}$$

$$\mathbf{F}(k) = \mathbf{\Phi}(k)\mathbf{e}(k) \in \mathbb{R}^{3N} \tag{41}$$

$$\overline{\mathbf{Q}} = diag(\mathbf{Q}, \mathbf{Q}, ..., \mathbf{Q}) \in \mathbb{R}^{3N} \times \mathbb{R}^{3N} \tag{42}$$

$$\overline{\mathbf{R}} = diag(\mathbf{R}, \mathbf{R}, ..., \mathbf{R}) \in \mathbb{R}^{2M} \times \mathbb{R}^{2M} \tag{43}$$

After some algebraic manipulation, one may show that the cost function may be rewritten as:

$$J(\hat{\mathbf{U}}_B) = \frac{1}{2} \hat{\mathbf{U}}_B^T \mathbf{H}_{qp}(k) \hat{\mathbf{U}}_B + \mathbf{f}_{qp}(k)^T \hat{\mathbf{U}}_B \tag{44}$$

where:

$$\mathbf{G}_n(k) = \overline{\mathbf{Q}}\mathbf{G}(k) \tag{45}$$

$$\mathbf{H}_{qp}(k) = 2(\mathbf{G}(k)^T \overline{\mathbf{Q}} \mathbf{G}(k) + \overline{\mathbf{R}}) \tag{46}$$

$$\mathbf{f}_{qp}(k) = 2\mathbf{G}_n(k)^T (\mathbf{F}(k) - \mathbf{E}_r) \tag{47}$$

Note that constants terms (i.e. not depending on $\hat{\mathbf{U}}_B$) may be dropped from $J(\hat{\mathbf{U}}_B)$, because they do not influence the optimization decision. Furthermore, define:

$$\mathbf{A}_{qp}(k) = diag(\mathbf{S}_B, \mathbf{S}_B, ..., \mathbf{S}_B) \in \mathbb{R}^{4M} \times \mathbb{R}^{2M} \tag{48}$$

$$\mathbf{b}_{qp}(k) = \begin{bmatrix} \mathbf{b}_B(k) \\ \mathbf{b}_B(k+1) \\ \vdots \\ \mathbf{b}_B(k+M-1) \end{bmatrix} \in \mathbb{R}^{4M} \tag{49}$$

Finally, the optimal control sequence is given by the solution to the following optimization problem:

$$\hat{\mathbf{U}}_B^* = \arg \min_{\hat{\mathbf{U}}_B \in \mathbb{R}^{2M}} \frac{1}{2} \hat{\mathbf{U}}_B^T \mathbf{H}_{qp}(k) \hat{\mathbf{U}}_B + \mathbf{f}_{qp}(k)^T \hat{\mathbf{U}}_B \quad \text{subjected to } \mathbf{A}_{qp}(k) \leq \mathbf{b}_{qp}(k) \tag{50}$$

which is a instance of the Quadratic Programming (QP) class of optimization problems (Gill et al., 1981; Coleman and Li, 1996). Despite we determine a control sequence, the command effectively applied at time step k is $\mathbf{u}_B^* = \hat{\mathbf{u}}_B^*(k|k)$.

6 Simulation Results

The simulation was developed using MATLAB/Simulink. The differential drive robot dynamics is simulated using the unicycle's non-linear model, shown in Equation (5). We model the vision system by adding gaussian noise with zero mean and $\mathbf{\Sigma}_m$ covariance matrix to the robot's kinematic state to simulate measurement noise. To solve the quadratic program of Equation (50), we use the `quadprog` MATLAB function. The trajectories used are Lissajous curves generated by the following equations:

$$x(t) = A_1 \sin(\omega_1 t + \delta) \tag{51}$$

$$y(t) = A_2 \sin(\omega_2 t) \tag{52}$$

where A_1, A_2, ω_1, ω_2 and δ are parameters. From Equations (51) and (52), we may derive $\psi_r(t)$ and $\mathbf{u}_r(t) = [v_r(t), \omega_r(t)]^T$ using Equations (8), (9) and (10). For all simulations, we used the parameters shown in Table 1.

Parameter	Value
ω_{MAX}	17 rad/s
r	3 cm
l	6 cm
T	1/30 s
A_1	1
A_2	1
δ	$\pi/2$
M	10
N	10

Table 1: Fixed parameters.

Additionally, we fixed $\mathbf{Q} = diag(4, 40, 0.1)$. Two matrices $\mathbf{R}$ were considered: $\mathbf{R}_1 = \mathbf{I}_{2\times 2}$ and $\mathbf{R}_2 = 0.002\mathbf{R}_1$. Matrix $\mathbf{R}_2$ has smaller weights, so the controller using $\mathbf{R}_2$ suffers less penalty for using control effort and we expect higher bandwidth. On the other hand, the lower bandwidth of the controller using $\mathbf{R}_1$ should make it less sensible to measurement noise.

For generating a trajectory, we made $\omega_1 = 3m$ and $\omega_2 = 2m$, then we used a numeric algorithm to search for the value of m that made $\max(\max(\omega_L(t)), \max(\omega_R(t)))$ close to $0.95\omega_{MAX}$. The idea was to come close to actuation saturation, while still lefting $0.05\omega_{MAX}$ for feedback control. Moreover, the robot's initial condition is a deviation from the reference trajectory of $\Delta\mathbf{q} = [\Delta x, \Delta y, \Delta\psi]^T = [0.1m, 0.05m, 0.05rad]^T$. When measurement noise was considered, we used $\mathbf{\Sigma}_1 = diag(0.04m, 0.04m, 0.05rad)^2$ as the measurement noise covariance matrix. Finally, the simulation time was 30 seconds for all simulations.

Four simulations cases were considered: 1) $\mathbf{\Sigma}_m = \mathbf{0}_{3\times 3}$ and $\mathbf{R} = \mathbf{R}_1$; 2) $\mathbf{\Sigma}_m = \mathbf{0}_{3\times 3}$ and $\mathbf{R} = \mathbf{R}_2$; 3) $\mathbf{\Sigma}_m = \mathbf{\Sigma}_1$ and $\mathbf{R} = \mathbf{R}_1$; and 4) $\mathbf{\Sigma}_m = \mathbf{\Sigma}_1$ and $\mathbf{R} = \mathbf{R}_2$. Figures 2, 3, 4, and 5 show reference and actual trajectories for each simulation case.

Observe that the we obtained expected results. When we do not have measurement noise, the controller using $\mathbf{R}_2$ is clearly superior: the robot quickly eliminates the initial tracking error, as may be seen in Figure 3. On the other hand, when measurement noise is present, the higher bandwidth of the controller using $\mathbf{R}_2$ makes it too sensible (see Figure 5), while the controller using $\mathbf{R}_1$ does not show a significant difference from the noiseless case (compare Figures 2 and 4). Finally, note in Figure 6 that, despite a control effort corrupted by measurement noise, the controller effectively respect the actuation constraint.

Regarding computational cost, the unoptimized MATLAB function that runs the whole MPC controller, including building the required matrices and solving the quadratic program, takes a mean processing time of 9 ms with standard deviation of 2 ms on a desktop computer with a Core i7-3770 3.40 GHz quad-core CPU and 16 GB of RAM. This performance is enough for controlling

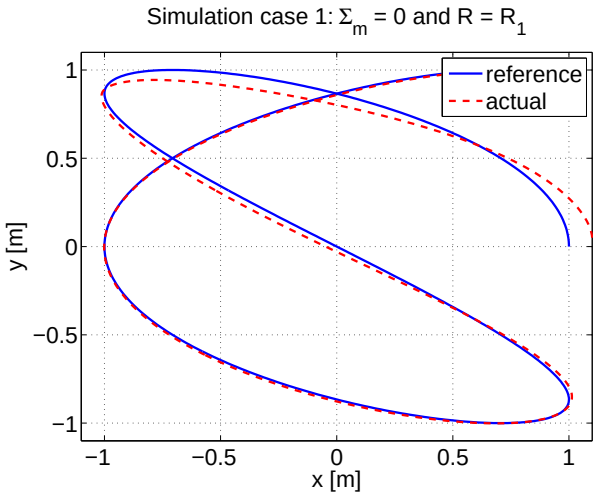

Figure 2: Simulation 1: $\mathbf{\Sigma}_m = \mathbf{0}_{3\times 1}$ and $\mathbf{R} = \mathbf{R}_1$.

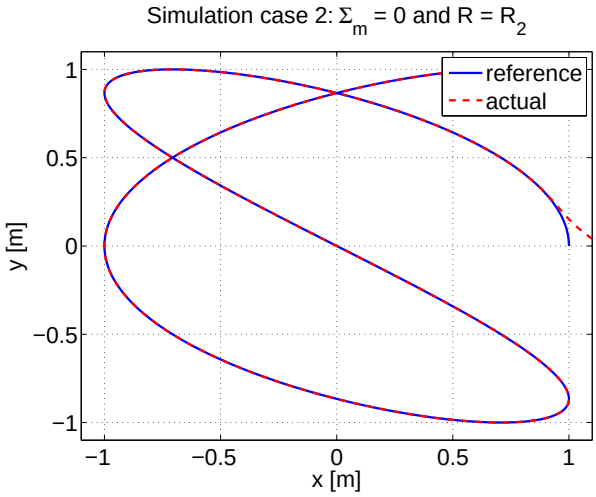

Figure 3: Simulation 2: $\mathbf{\Sigma}_m = \mathbf{0}_{3\times 1}$ and $\mathbf{R} = \mathbf{R}_2$.

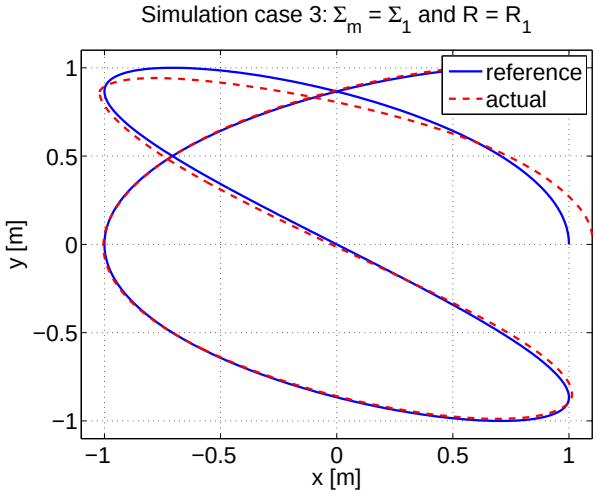

Figure 4: Simulation 3: $\mathbf{\Sigma}_m = \mathbf{\Sigma}_1$ and $\mathbf{R} = \mathbf{R}_1$.

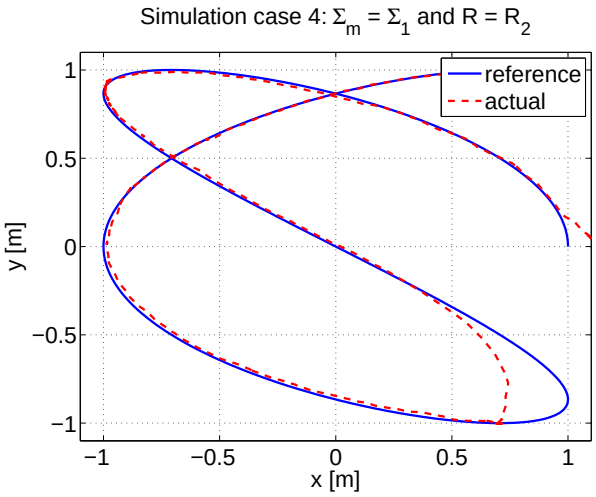

Figure 5: Simulation 4: $\mathbf{\Sigma}_m = \mathbf{\Sigma}_1$ and $\mathbf{R} = \mathbf{R}_2$.

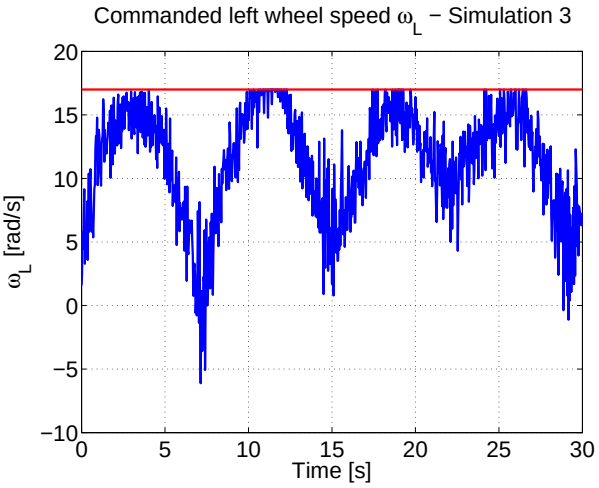

Figure 6: Commanded left wheel's angular speed ω_L during simulation case 3 ($\mathbf{\Sigma}_m = \mathbf{\Sigma}_1$ and $\mathbf{R} = \mathbf{R}_1$).

a single robot at 30 Hz, but it is not sufficient for robot soccer, where a team of robots must be controlled simultaneously and processing power is shared with other algorithms. However, we expect that an optimized implementation in C++ will greatly reduce this processing time.

7 Conclusions

Despite the challenges present in the trajectory tracking by a differential drive robot problem, the model predictive controller developed in this work achieves high performance, while effectively handling the actuation constraints imposed by each wheel's speed limits. Simulation results have shown that a compromise between state tracking performance and sensitivity to noise may be achieved by manipulating the weight matrices $\mathbf{Q}$ and $\mathbf{R}$. We should add that direct vision measurement was used as feedback, thus using a state observer in conjunction with the proposed controller will probably yield improved performance.

As a future research direction, we are looking to incorporate other issues in our MPC formulation, such as wheel dynamics, computing delay, and obstacle avoidance. Moreover, we expect to finish this controller implementation in our real robot system soon, thus allowing an evaluation using experiments with real robots.

Furthermore, more analysis considering how the controller's parameters affect its behavior are desireable. For example, increasing M and N usually yield better tracking performance at the expense of a higher computational cost. A quantitative analysis of this trade-off will provide useful insight for selecting better values for M and N.

Acknowledgements

The author would like to thank ITAndroids' sponsors: CESAR, Poliedro and Mectron Odebrecht.

References

7th Latin American IEEE Student Robotics Competition (2008). Rules for the IEEE Very Small Competition.

Camacho, E. F. and Alba, C. B. (2007). *Model Predictive Control*, 2ª edn, Springer.

Coleman, T. F. and Li, Y. (1996). A Reflective Newton Method for Minimizing a Quadratic Function Subject to Bounds on some of the Variables, **6**(4): 1040–1058.

Gill, P. E., Murray, W. and Wright, M. H. (1981). *Practical Optimization*, Academic Press.

Kanjanawaniskul, K. (2012). Motion Control of a Wheeled Mobile Robot Using Model Predictive Control: A Survey, *KKU Research Journal* **17**(5): 811–837.

Klančar, G. and Škrjanc, I. (2007). Tracking-error model-based predictive control for mobile robots in real time, *Robotics and Autonomous Systems* **55**: 460–469.

Kühne, F., Lages, W. F. and da Silva Jr., J. M. G. (2005). Mobile robot trajectory tracking using model predictive control, *Anais do 7º Simpósio Brasileiro de Automação Inteligente*, Sociedade Brasileira de Automática.

Siegwart, R., Nourbakhsh, I. R. and Scaramuzza, D. (2011). *Autonomous Mobile Robots*, 2ª edn, The MIT Press.