REAL-TIME CONTROL OF A MOBILE ROBOT USING LINEARIZED MODEL PREDICTIVE CONTROL

Walter Fetter Lages *
Jorge Augusto Vasconcelos Alves *

* *Federal University of Rio Grande do Sul*
Electrical Engineering Department
Av.Osvaldo Aranha, 103
90035-190 Porto Alegre, RS, Brazil
{fetter,jorge}@ece.ufrgs.br

Abstract: This paper proposes an optimal control strategy for a differential-drive mobile robot. It is well known that such a system can not be feedback stabilized by a smooth time-invariant control law. By using model predictive control (MPC), an appropriate control law is implicitly obtained. Furthermore, the system physical constraints on state and inputs are dealt with in a straightforward way. The optimization problem is solved by quadratic programming (QP) and experimental results show that the control law computation can be performed under the system real-time requirements.

Keywords: Real-time systems, Predictive control, Mobile robots, Quadratic programming

1. INTRODUCTION

Despite the apparent simplicity of the kinematic model of a mobile robot, the design of stabilizing control laws for those systems is a considerable challenge. Due to Brockett conditions (Brockett, 1982), a continuously differentiable, smooth stabilizing feedback control law can not be obtained. To overcome these limitations non-smooth and time-varying control laws have been proposed (Bloch and McClamroch, 1989; Samson and Ait-Abderrahim, 1991; Canudas de Wit and Sørdalen, 1992; McCloskey and Murray, 1997). Recent works dealing with robust and adaptive control of mobile robots can be found in Oya *et al.* (2003) and Dixon *et al.* (2004).

However, in realistic implementations it is difficult to obtain good performance, due to the constraints on inputs or states that naturally arise. Usually, the controller is designed under the assumption that there are no limitations on inputs or state and at most the actual restrictions are take into account afterwards. By using model predictive control (MPC), restrictions can be handled in a straightforward way while generating the control law. For a mobile robot this is an important issue, since the position of the robot can be restricted to belong to a safe region of operation and control actions that respect actuators limits can be generated.

Also, coordinate transformations of the dynamic system to chained or power forms (Bloch and McClamroch, 1989) are not necessary anymore, which turns the choice of tuning parameters for the MPC more intuitive.

For complex, constrained, multivariable control problems, MPC has become an accepted standard

in the process industries (Bemporad *et al.*, 2002). Typically, it is used when the plant is sufficiently slow to allow its implementation (Mayne *et al.*, 2000). However, for systems with fast and/or nonlinear dynamics, the implementation of such technique remains fundamentally limited in its applicability, due to the large amount of on-line computation that is required (Cannon and Kouvaritakis, 2000).

Model predictive control of mobile robots are recent and sparse (Ollero and Amidi, 1991; Essen and Nijmeijer, 2001). A possible cause is that the model of a mobile robot is nonlinear and although nonlinear model predictive control (NMPC) has been developed (Mayne *et al.*, 2000; Allgöwer *et al.*, 1999; Chen and Allgöwer, 1998), the computational effort is much higher than the linear version. In NMPC the nonlinear programming problem to be solved on-line is nonconvex and the number of decision variables is larger than its linear counterpart. Hence, a global minimum is in general very hard to find (Henson, 1998). In this paper, we propose a strategy to overcome some of the problems related to the use o model predictive control for nonlinear systems. The fundamental idea consists in using a successive linearization approach, yielding a linear, time-varying description of the system that can be controlled through linear MPC. Then, by considering the control inputs as the decision variables, it is possible to transform the optimization problem in a Quadratic programming (QP) problem. Since that is a convex problem, the QP problem can be solved by numerically robust solvers, leading to global optimal solutions.

In a previous paper (Kühne *et al.*, 2004), authors have showed that the computational effort needed to run the proposed MPC algorithm is such that a real-time implementation is possible. However, that results were based on the number of flops (as reported by Matlab) needed to compute the control law and an estimate of processor performance in Mflops. In this paper the viability of the approach is demonstrated by an actual real-time implementation, with timing measures taken on the working system.

2. MODEL OF THE MOBILE ROBOT

The kinematic model of the mobile robot (see Fig. 1) is given by (1) (Campion *et al.*, 1996):

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = w \end{cases} \tag{1}$$

or, in a more compact form as

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}), \tag{2}$$

where $\mathbf{x} \triangleq [x \;\; y \;\; \theta]^T$ describes the configuration (position and orientation) of the center of the axis of the wheels, C, with respect to a global inertial frame $\{O, X, Y\}$. $\mathbf{u} \triangleq [v \;\; w]^T$ is the control input, where v and w are the linear and the angular velocities, respectively.

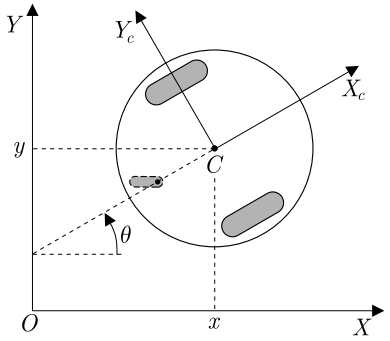

Fig. 1. Coordinate system of the mobile robot.

A linear model is obtained by computing an error model with respect to a reference car described by (2). Hence, the reference trajectory $\mathbf{x}_r$ and $\mathbf{u}_r$ are related by:

$$\dot{\mathbf{x}}_r = f(\mathbf{x}_r, \mathbf{u}_r) \tag{3}$$

By expanding the right side of (2) in Taylor series around the point $(\mathbf{x}_r, \mathbf{u}_r)$ and discarding the high order terms it follows that

$$\dot{\mathbf{x}} = f(\mathbf{x}_r, \mathbf{u}_r) + \left.\frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}}\right|_{\substack{\mathbf{x}=\mathbf{x}_r \\ \mathbf{u}=\mathbf{u}_r}} (\mathbf{x} - \mathbf{x}_r) + \left.\frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}}\right|_{\substack{\mathbf{x}=\mathbf{x}_r \\ \mathbf{u}=\mathbf{u}_r}} (\mathbf{u} - \mathbf{u}_r), \tag{4}$$

or

$$\dot{\mathbf{x}} = f(\mathbf{x}_r, \mathbf{u}_r) + f_{\mathbf{x}_r}(\mathbf{x} - \mathbf{x}_r) + f_{\mathbf{u},r}(\mathbf{u} - \mathbf{u}_r), \tag{5}$$

where $f_{\mathbf{x}_r}$ and $f_{\mathbf{u}_r}$ are the jacobians of f with respect to $\mathbf{x}$ and $\mathbf{u}$, respectively, evaluated around the reference point $(\mathbf{x}_r, \mathbf{u}_r)$.

Then, the subtraction of (3) from (5) results in:

$$\dot{\tilde{\mathbf{x}}} = f_{\mathbf{x}_r}\tilde{\mathbf{x}} + f_{\mathbf{u}_r}\tilde{\mathbf{u}} \tag{6}$$

Hence, $\tilde{\mathbf{x}} \triangleq \mathbf{x} - \mathbf{x}_r$ represents the error with respect to the reference car and $\tilde{\mathbf{u}} \triangleq \mathbf{u} - \mathbf{u}_r$ is its associated perturbation control input.

The approximation of $\dot{\mathbf{x}}$ by using forward differences gives the following discrete-time system model:

$$\tilde{\mathbf{x}}(k+1) = \mathbf{A}(k)\tilde{\mathbf{x}}(k) + \mathbf{B}(k)\tilde{\mathbf{u}}(k), \tag{7}$$

with

$$\mathbf{A}(k) \triangleq \begin{bmatrix} 1 & 0 & -v_r(k)\sin\theta_r(k)T \\ 0 & 1 & v_r(k)\cos\theta_r(k)T \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{B}(k) \triangleq \begin{bmatrix} \cos\theta_r(k)T & 0 \\ \sin\theta_r(k)T & 0 \\ 0 & T \end{bmatrix}$$

where T is the sampling period and k is the sampling interval.

In Bloch and McClamroch (1989) it is shown that the nonlinear, nonholonomic system (1) is fully controllable, i.e., it can be steered from any initial state to any final state in finite time by using finite inputs. It is easy to see that when the robot is not moving, the linearization about a stationary operating point is not controllable. However, this linearization becomes controllable as long as the control input $\mathbf{u}$ is not zero (Samson and Ait-Abderrahim, 1991). This implies the tracking of a reference trajectory being possible with linear MPC (Essen and Nijmeijer, 2001).

3. THE MPC ALGORITHM

The essence of a MPC scheme is to optimize predictions of process behavior over a sequence of future control inputs. Such a prediction is accomplished by using a process model over a finite time interval, called the *prediction horizon*. At each sampling time, the model predictive controller generates an optimal control sequence by solving an optimization problem. The first element of this sequence is applied to the plant. The problem is solved again at the next sampling time using the updated process measurements and a shifted horizon.

The objective function to be minimized is:

$$\Phi(k) = \sum_{j=1}^{N} \tilde{\mathbf{x}}^T(k+j|k)\mathbf{Q}\tilde{\mathbf{x}}(k+j|k) + \\ + \tilde{\mathbf{u}}^T(k+j-1|k)\mathbf{R}\tilde{\mathbf{u}}(k+j-1|k), \quad (8)$$

where N is the prediction horizon and $\mathbf{Q} \geq 0$, $\mathbf{R} > 0$ are weighting matrices. The notation $a(m|n)$ indicates the value of a at the instant m predicted at instant n.

Hence, the optimization problem can be stated as to find $\tilde{\mathbf{u}}^\star$ such that:

$$\tilde{\mathbf{u}}^\star = \arg\min_{\tilde{\mathbf{u}}} \{\Phi(k)\} \quad (9)$$

The problem of minimizing (8) is solved at each time step k, yielding a sequence of optimal control $\{\tilde{\mathbf{u}}^\star(k|k), \cdots, \tilde{\mathbf{u}}^\star(k+N-1|k)\}$ and the optimal cost $\Phi^\star(k)$. The MPC control law is implicitly given by the first control action of $\tilde{\mathbf{u}}^\star(k|k)$. A block diagram of the system is shown in Fig. 2.

To recast the optimization problem in a usual quadratic programming form, we introduce the following vectors:

$$\bar{\mathbf{x}}(k+1) \triangleq \begin{bmatrix} \tilde{\mathbf{x}}(k+1|k) \\ \tilde{\mathbf{x}}(k+2|k) \\ \vdots \\ \tilde{\mathbf{x}}(k+N|k) \end{bmatrix} \quad \bar{\mathbf{u}}(k) \triangleq \begin{bmatrix} \tilde{\mathbf{u}}(k|k) \\ \tilde{\mathbf{u}}(k+1|k) \\ \vdots \\ \tilde{\mathbf{u}}(k+N-1|k) \end{bmatrix}$$

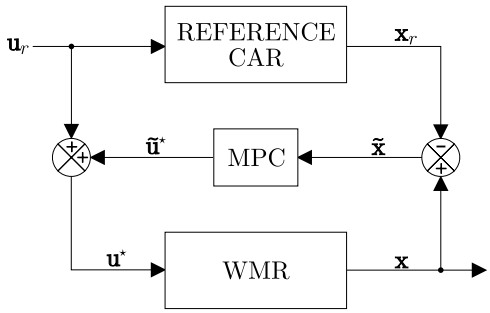

Fig. 2. Block diagram of the LMPC.

Thus, (8) can be rewritten as:

$$\Phi(k) = \bar{\mathbf{x}}^T(k+1)\bar{\mathbf{Q}}\bar{\mathbf{x}}(k+1) + \bar{\mathbf{u}}^T(k)\bar{\mathbf{R}}\bar{\mathbf{u}}(k), \quad (10)$$

with

$$\bar{\mathbf{Q}} \triangleq \begin{bmatrix} \mathbf{Q} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{Q} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{Q} \end{bmatrix} \quad \bar{\mathbf{R}} \triangleq \begin{bmatrix} \mathbf{R} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{R} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{R} \end{bmatrix}$$

Therefore, it is possible from (7) to write $\bar{\mathbf{x}}(k+1)$ as:

$$\bar{\mathbf{x}}(k+1) = \bar{\mathbf{A}}(k)\tilde{\mathbf{x}}(k|k) + \bar{\mathbf{B}}(k)\bar{\mathbf{u}}(k), \quad (11)$$

with

$$\bar{\mathbf{A}}(k) \triangleq \begin{bmatrix} \alpha(k+1,k) \\ \alpha(k+2,k) \\ \vdots \\ \alpha(k+N,k) \end{bmatrix}$$

and

$$\bar{\mathbf{B}}(k) \triangleq \begin{bmatrix} \beta_{11}(k) & \mathbf{0} & \cdots & \mathbf{0} \\ \beta_{21}(k) & \beta_{22}(k) & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{N1}(k) & \beta_{N2}(k) & \cdots & \beta_{NN}(k) \end{bmatrix}$$

where β_{ij} and $\alpha(k,j)$ are defined as:

$$\beta_{ij}(k) \triangleq \alpha(k+i, k+j)\mathbf{B}(k+j-1)$$

$$\alpha(k_1,k_0) \triangleq \begin{cases} \mathbf{I} & \text{if } k_1 = k_0 \\ \prod_{i=0}^{n-1} \mathbf{A}(k+i) & \text{if } k_1 > k_0. \end{cases}$$

From (10) and (11), we can rewrite the objective function (8) in a standard quadratic form:

$$\Phi(k) = \frac{1}{2}\bar{\mathbf{u}}^T(k)\mathbf{H}(k)\bar{\mathbf{u}}(k) + \mathbf{f}^T(k)\bar{\mathbf{u}}(k) + \mathbf{d}(k)$$

with

$$\mathbf{H}(k) \triangleq 2\left(\bar{\mathbf{B}}(k)^T(k)\bar{\mathbf{Q}}\bar{\mathbf{B}}(k) + \bar{\mathbf{R}}\right)$$
$$\mathbf{f}(k) \triangleq 2\bar{\mathbf{B}}^T(k)\bar{\mathbf{Q}}\bar{\mathbf{A}}(k)\tilde{\mathbf{x}}(k|k)$$
$$\mathbf{d}(k) \triangleq \tilde{\mathbf{x}}^T(k|k)\bar{\mathbf{A}}^T(k)\bar{\mathbf{Q}}\bar{\mathbf{A}}(k)\tilde{\mathbf{x}}(k|k)$$

The matrix $\mathbf{H}$ is a *Hessian* matrix, and must be positive definite. It describes the quadratic part of the objective function, and the vector $\mathbf{f}$ describes the linear part. $\mathbf{d}$ is independent of $\tilde{\mathbf{u}}$ and does not matter for the determination of $\mathbf{u}^\star$.

Model predictive control is based on the assumption that for a small time horizon plant and model behavior are the same. For this assumption to hold the plant/model mismatch should be kept small. Obviously, for any real world plant, control inputs are subject to physical limitations. Hence, to avoid large plant/model mismatch those limitations should be considered while computing control inputs. This can be done in a straightforward way by defining upper and lower bounds on the control input. The optimization problem must then be solved while ensuring that the control will remain between certain lower and upper bounds. The control constraint can be written as:

$$\mathbf{u}_{min}(k) \leq \mathbf{u}(k) \leq \mathbf{u}_{max}(k), \tag{12}$$

where the subscripts *min* and *max* stands for lower and upper bounds, respectively.

Hence, the optimization problem in (9) can be reformulated as to find $\tilde{\mathbf{u}}^{\star}$ such that:

$$\tilde{\mathbf{u}}^{\star} = \arg\min_{\tilde{\mathbf{u}}} \{\Phi(k)\} \tag{13}$$

s. a.

$$\mathbf{D}\tilde{\mathbf{u}} \leq \mathbf{d} \tag{14}$$

where $\Phi(k)$ is the objective function and $\tilde{\mathbf{u}}$ is the free variable in the optimization. Inequality (14) is a general way to describe constraints in the control variables. For instance, when there are just control amplitude constraints as in (12), we have

$$\begin{bmatrix} \mathbf{I} \\ -\mathbf{I} \end{bmatrix} \tilde{\mathbf{u}} \leq \begin{bmatrix} \tilde{\mathbf{u}}_{max} \\ -\tilde{\mathbf{u}}_{min} \end{bmatrix},$$

which turns out that

$$\tilde{\mathbf{u}}_{min} \leq \tilde{\mathbf{u}} \leq \tilde{\mathbf{u}}_{max}$$

Since the free variable in the optimization is $\tilde{\mathbf{u}}(k)$, the constraint (12) must be rewritten with respect to this variable:

$$\mathbf{u}_{min}(k) - \mathbf{u}_r(k) \leq \tilde{\mathbf{u}}(k) \leq \mathbf{u}_{max}(k) - \mathbf{u}_r(k),$$

4. REAL-TIME IMPLEMENTATION

Figure 3 shows the mobile robot developed in our labs and used in this work. It has a cylindrical geometry with 1.35m in height and 0.30 cm in radius and uses a differential-drive steering. The software implementing the linearized MPC controller was written in C++. To ensure accurate timing, a hard real-time extension to the Linux kernel called RTAI (Dozio and Mantegazza, 2003) was used. The optimization problem was solved by using the OOQP library (Gertz and Wright, 2003).

The initial configuration of the mobile robot and the reference car are $\mathbf{x}(0) = [0 \quad -1 \quad \pi/2]^T$ and $\mathbf{x}_r(0) = [0 \;\; 0 \;\; 0]^T$, respectively. The weighting matrices are $\mathbf{Q} = diag(10, 10, 0.5)$ and $\mathbf{R} = 0.1\mathbf{I}_{2\times 2}$. The prediction horizon is $N = 5$. Constraints in

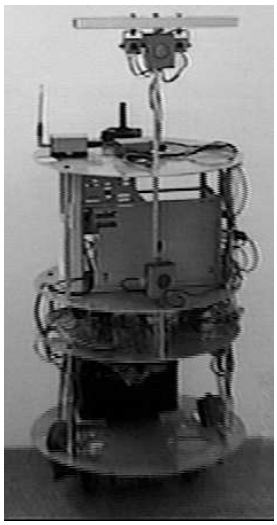

Fig. 3. The Twil mobile robot.

the amplitude of the control variables are: $v_{min} = -0.47m/s$, $v_{max} = 0.47m/s$, $w_{min} = -3.3rad/s$ and $w_{max} = 3.3rad/s$.

It can be clearly seen from Fig. 4 that the state asymptotically converges to the reference. It should be noted that, due to initial orientation, the mobile robot have to turn away from the reference trajectory to cope with the nonholonomic constraints. In Fig. 5, it can be seen that the control inputs are inside the limits imposed by the constraints.

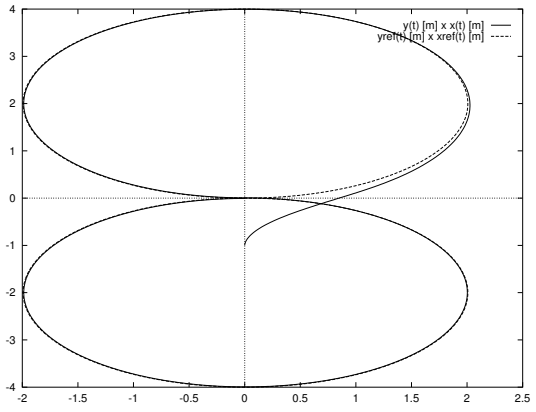

Fig. 4. Reference and actual trajectory.

Figure 6 shows the time needed to compute the control law as a function of time for three different processors. It can be seen that even for an obsolete processor such as a Pentium II 300MHz, the control law can be computed within the sampling period. The variations on the computing time are in part due to the optimization process, but mainly due to jitter in the scheduling of the task that computes the control. The causes of the jitter are the other (non real-time) tasks executed by the processor.

Table 1 shows some results relating the computational cost as a function of the prediction

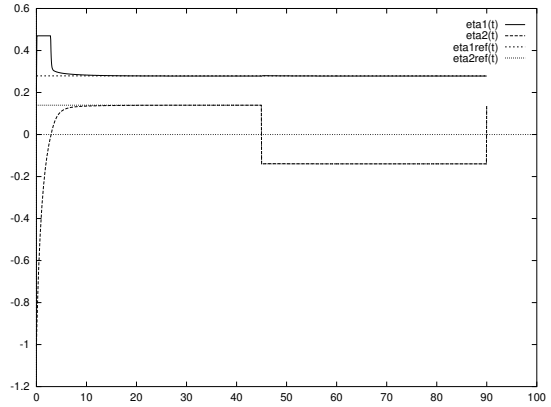

Fig. 5. Control inputs.

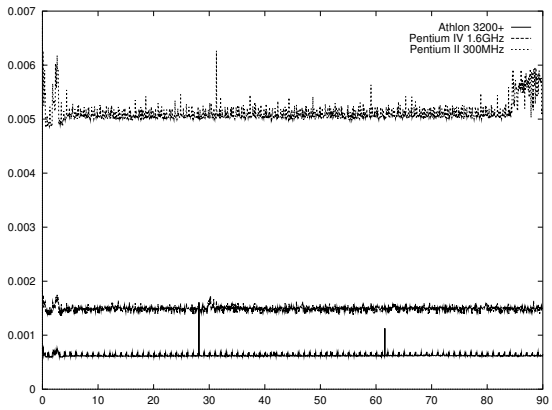

Fig. 6. Control law computing time.

horizon N: the average number of floating-point operations (flo) used to compute the control law at each sampling time, its estimated computing time and its actual computing time and the actual Mflops delivered by the processor. The estimated computing time was obtained considering a performance of 282.5Mflops for a Pentium IV 1.6GHz processor, according to Aburto (1992). The actual computing time was measured by using the `rt_get_time_ns()` function from RTAI system.

Table 1. Prediction horizon influence on computing time.

N	flo	Computing time (s)		Mflops
		Estimated	Actual	
1	4343	0.000015374	0.00058610	8.9344
3	9529	0.000033733	0.00098337	9.6901
5	25643	0.000090777	0.0015	17.0953
10	160180	0.00056704	0.0040	40.0450
15	528570	0.0019	0.0094	56.2309
20	1269500	0.0045	0.0188	67.5266
30	4949000	0.0175	0.0536	92.3321

Indeed, the data in Table 1 provides enough evidence that a standard of the shelf computer is able to run a MPC based controller for a mobile robot. The MPC algorithm proposed here could be computed for $N = 20$ in about 20ms, while the dynamics of the mobile robot used here is such that sampling periods in the order of 50ms are adequate, revealing that a real-time implementation is plenty possible.

Obviously the prediction horizon must be chosen in a way such that the computing time is smaller than the sampling period. Here, $T = 50ms$, therefore for $N = 30$ or above the MPC is not feasible. However, from Kühne *et al.* (2004) it is known that the current problem there is no sensible improvements on the state convergence rate for $N > 5$. Since for $N = 5$ the computing time is approximately 33 times smaller than the sampling period, it is a good choice for the prediction horizon.

From Table 1 it is also clear that the number of floating point operations and the processor performance figures in Mflops do not provide a good estimate for the processing time, since the actual processing times were much longer than the estimated ones. One reason is that published or measured processor performances are typically peak performances that are not sustained for real applications where floating-point operations are mixed with integer operations in different patterns. Another reason is that in an actual real-time implementation of the proposed MPC strategy, some of the computing time is spent in housekeeping: allocating and deallocating memory for variables, dereferencing pointers, accessing hardware registers, real-time scheduling, etc. Such operations do not count as floating-point operations, but take some time to execute. On the other hand, they are related to the problem setup and not to how many steps it takes to solve the optimization problem. Hence they show up as a approximately constant time overhead at each sampling time. That explains why for larger values of N the measured floating-point performance improves: the housekeeping overhead remains constant while the floating-point load increases, therefore, a larger portion of the time is spent with floating-point processing.

5. CONCLUSION

This paper presented an application of MPC to the problem of trajectory tracking of a nonholonomic mobile robot. The solution of the optimization problem through a standard QP method was shown. The obtained control signals were such that the constraints imposed on the control variables were respected.

As shown above, the choice of MPC for the application given here is well justified by some advantages: the straightforward way in which state/input constraints can be handled; coordinate transformations to a chained or power form are not necessary; the MPC implicitly generates

a control law that thus deals with Brockett conditions.

The viability of the proposed scheme was demonstrated by a real-time implementation, and performance data showed that even low-end or obsolete processors have enough processing power to meet the timing requirements.

ACKNOWLEDGMENTS

Authors ar partially supported by CAPES (Coordenação de Aperfeçoamento de Pessoal de Nível Superior) and FAPERGS (Fundação de Amparo a Pesquisa do Estado do Rio Grande do Sul).

REFERENCES

Aburto, Al (1992). FLOPS C program (double precision) v2.0 18 dec 1992. ¡ftp://ftp.nosc.mil/pub/aburto/flops¿.

Allgöwer, F., T. A. Badgwell, J. S. Qin, J. B. Rawlings and S. J. Wright (1999). Nonlinear predictive control and moving horizon estimation: an introductory overview. In: *Advances in Control: Highlights of ECC'99* (P. M. Frank, Ed.). pp. 391–449. Springer-Verlag. New York.

Bemporad, A., M. Morari, V. Dua and E. N. Pistikopoulos (2002). The explicit linear quadratic regulator for constrained systems. *Automatica* **38**(1), 3–20.

Bloch, A. M. and N. H. McClamroch (1989). Control of mechanical systems with classical nonholonomic constraints. In: *Proceedings of the 28th IEEE American Conference on Decision and Control*. Tampa, FL. pp. 201–205.

Brockett, R. W. (1982). *New Directions in Applied Mathematics*. Springer-Verlag. New York.

Campion, G., G. Bastin and B. D'Andréa-Novel (1996). Structural properties and classification of kinematic and dynamical models of wheeled mobile robots. *IEEE Transactions on Robotics and Automation* **12**(1), 47–62.

Cannon, M. and B. Kouvaritakis (2000). Continuous-time predictive control of constrained nonlinear systems. In: *Nonlinear Model Predictive Control* (F. Allgöwer and A. Zeng, Eds.). Vol. 26 of *Progress in Systems and Control Theory*. pp. 205–215. Birkhäuser Verlag AG. Basel, Switzerland.

Canudas de Wit, C. and O. J. Sørdalen (1992). Exponential stabilization of mobile robots with nonholonomic constraints. *IEEE Transactions on Automatic Control* **37**(11), 1791–1797.

Chen, C. C. and F. Allgöwer (1998). A quasi-infinite horizon nonlinear nodel predictive control with guaranteed stability. *Automatica* **34**(10), 1205–1217.

Dixon, W. E., M. S. de Queiroz, D. M. Dawson and T. J. Flynn (2004). Adaptive tracking and regulation of a wheeled mobile robot with controller/update law modularity. *IEEE Control Systems Technology* **12**(1), 138–147.

Dozio, L. and Paulo Mantegazza (2003). Linux real time application interface (RTAI) in low cost high performance motion control. In: *Proceedings of the Motion Control 2003 Conference*. Associazione Nazionale Italiana per l'Automazione. Milano, Italy.

Essen, H. V. and H. Nijmeijer (2001). Nonlinear model predictive control of constrained mobile robots. In: *Proceedings of the European Control Conference*. Porto, Portugal. pp. 1157–1162.

Gertz, E. Michael and Stephen J. Wright (2003). Object-oriented software for quadratic programming. *ACM Transactions on Mathematical Software* **29**(1), 58–81.

Henson, M. A. (1998). Nonlinear model predictive control: current status and future directions. *Computers and Chemical Engineering* **23**(2), 187–202.

Kühne, Felipe, Walter Fetter Lages and João Manoel Gomes da Silva Jr. (2004). Model predictive control of a mobile robot using linearization. In: *Proceedings of the Mechatronics and Robotics 2004*. Aachen, Germany. pp. 525–530.

Mayne, D. Q., J. B. Rawlings, C. V. Rao and P. O. M. Scokaert (2000). Constrained model predictive control: stability and optimality. *Automatica* **36**(6), 789–814.

McCloskey, R. T. and R. M. Murray (1997). Exponential stabilization of driftless control systems using homogeneous feedback. *IEEE Transactions on Automatic Control* **42**(5), 614–628.

Ollero, A. and O. Amidi (1991). Predictive path tracking of mobile robots: application to the CMU Navlab. In: *Proceedings of the 5th IEEE International Conference on Advanced Robotics*. Pisa, Italy. pp. 1081–1086.

Oya, M., C. H. Yu and R. Katoh (2003). Robust adaptive motion/force tracking control of uncertain nonholonomic mechanical systems. *IEEE Transactions on Robotics and Automation* **19**(1), 175–181.

Samson, C. and K. Ait-Abderrahim (1991). Feedback control of a nonholonomic wheeled cart in cartesian space. In: *Proceedings of the IEEE International Conference on Robotics and Automation*. Sacramento, CA. pp. 1136–1141.