Robotica (2009) volume 27, pp. 281–290. © 2008 Cambridge University Press
doi:10.1017/S0263574708004700 Printed in the United Kingdom

Performance-based reactive navigation for non-holonomic mobile robots

Michael Defoort†*, Jorge Palos‡, Annemarie Kokosy‡§, Thierry Floquet§ and Wilfrid Perruquetti§

†*Department of System Design Engineering, Keio University, Yokohama, Japan.*
‡*ISEN, 41 bvd Vauban, 59 046 Lille Cedex, France.*
§ *LAGIS UMR CNRS 8146, Ecole Centrale de Lille, BP 48, Cité Scientifique, 59651 Villeneuve-d'Ascq, France.*

(Received in Final Form: April 28, 2008. First published online: June 4, 2008)

SUMMARY
This paper presents an architecture for the navigation of an autonomous mobile robot evolving in environments with obstacles. Instead of addressing the motion planning and control problems in different contexts, these issues are described in connected modules with performance requirement considerations. The path planning problem is formulated as a constrained receding horizon planning problem and is solved in real time with an efficient computational method that combines non-linear control theory, B-spline basis function and non-linear programming. An integral sliding mode controller is used for trajectory tracking. The closed-loop stability of the tracking errors is guaranteed in spite of unknown disturbances. It is also shown that this strategy is particularly useful if integral sliding mode control is combined with other methods to further robustify against perturbations. The effectiveness, perfect performance of obstacle avoidance, real time and high robustness properties are demonstrated by experimental results.

KEYWORDS: Non-holonomic mobile robots; Reactive navigation; Receding horizon planning; Sliding mode control.

1. Introduction

Wheeled mobile robots (WMRs) have been widely studied in the last two decades due to their practical importance and theoretically interesting properties. Indeed, there are considerable research efforts towards solving mobile robot navigation in different applications in indoor and outdoor environments (see refs. [1, 2] and survey paper [3]). For some navigation tasks like planetary exploration, robots are required to travel long distances within constrained resources (energy, time, etc.). In such cases, efficient motion planning and control algorithms are needed in order to achieve the goal while meeting some performance issue, such as geometric-based or time-based criterion.

Although motion planning and control design are closely related in the robot navigation problem, they are usually addressed as two separate problems in much of the existing literature. Motion planning consists in generating a collision-free trajectory from the initial to the final desired positions and control design is the determination of the physical inputs to the robot motion components. These two problems are typically solved using methods from different areas such as those in artificial intelligence and control theory. Such a separation makes it difficult to address robot performances in a complete application, since the planned trajectory may not be efficiently tracked. This fact can imply that the meaning of optimization in each step is lost. For instance, in a typical time optimal trajectory planning, the open-loop control schemes result in bang–bang or bang–singular–bang controls.[4] However, the discontinuities of the planned open-loop control may not produce a satisfactory path tracking result in practice and may not be applicable to high speed travelling. In this research, we bridge the gap between trajectory planning and motion control.

Many theoretically challenging properties stem from the so-called non-holonomic constraints imposed by the rolling wheels. A survey of non-holonomic control problems can be found in ref. [5]. Obstacles to the tracking of such systems are the uncontrollability of their linear approximation and the fact that the Brockett's necessary condition to the existence of a smooth time-invariant state feedback is not satisfied.[6] To overcome these difficulties, various methods have been investigated: homogeneous and time-varying feedbacks,[7,8] sinusoidal and polynomial controls,[9] piecewise continuous controls,[10] backstepping approaches,[11] hybrid controls[12] or discontinuous controls.[13,14] However, most of these methods do not provide fast convergence and good robustness properties. Most of the control laws ensuring exponential or finite time convergence[15] are known to be non-robust under disturbances or modelling errors. On the other hand, control design methodologies, like smooth time-varying feedback,[8] are quite insensitive to perturbations but imply a slow convergence.

In this paper, we propose a practical scheme for real-time motion planning and control of non-holonomic mobile robot moving in an uncertain environment. As illustrated in Fig. 1, the scheme consists of two main parts: (i) a real-time collision-free motion planner; (ii) a trajectory tracking controller. For each module, we explicitly address the performance considerations. In implementation, the motion

* Corresponding author. E-mail: defoort@sum.sd.keio.ac.jp

https://doi.org/10.1017/S0263574708004700 Published online by Cambridge University Press

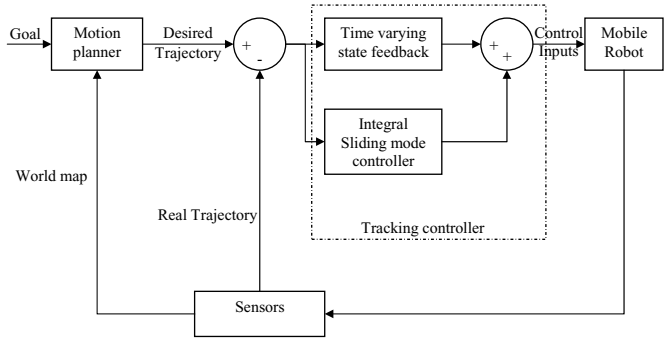


Fig. 1. Block diagram for navigation of a non-holonomic mobile robot with a motion planner and a sliding mode controller.

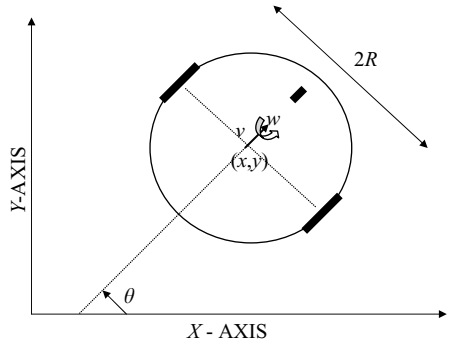


Fig. 2. Unicycle-type mobile robot.

planner dynamically generates the optimal trajectory while the robot runs. High precision motion tracking is achieved by the combination of integral sliding mode control and time-varying state feedback. The main results are general and can be applied whenever the integral sliding mode control is combined with other techniques to further robustify against disturbances. Experimentations support the validity of the theoretical analysis and show that the performance of a time-varying state feedback controller can be increased by this particular strategy.

The outline of this paper is as follows. In Section 2, the navigation problem is defined and the robot's non-holonomic model is described. Motion planning is discussed in Section 3. Trajectory tracking strategy is presented in Section 4. Finally, in Section 5, the integration of the different modules and the experimental results on a mobile robot are presented.

2. Problem Statement

2.1. Modelling of the mobile robot

The mobile robot, shown in Fig. 2, is of unicycle-type. The robot body is of symmetric shape and the mass centre is at the geometric centre C of the body. It has two driving wheels fixed to the axis which passes through C and one passive centred orientable wheel. The two fixed wheels of radius r, separated by $2R$, are independently controlled by two actuators (DC motors) and the passive wheel prevents the robot from tipping over as it moves on a plane. In this paper, it is assumed that the motion of the passive wheel can be ignored in the dynamics of the mobile robot. The centre of mass C, whose coordinates are (x, y), is located at the intersection of a straight line passing through the middle of the vehicle and the axis of the two driving wheels. The configuration of the robot can be described by

$$q = [x, y, \theta]^T,$$

where θ is its orientation in the global frame.

In this paper, the kinematics of the WMR are given under the non-holonomic constraints (see ref. [16] for details). The pure rolling and non-slipping non-holonomic conditions are described by

$$A^T(q)\dot{q} = 0 \quad \text{with} \quad A^T(q) = [-\sin\theta \ \cos\theta \ 0].$$

The kinematic equations can be written as follows:

$$\dot{q} = f(q)U, \tag{1}$$

where

$$f(q) = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix}, U = [v, w]^T$$

is the control input, v and w are the linear and angular velocities, respectively. The relationship between $[v, w]^T$ and the left and right velocities $[w_{\text{left}}, w_{\text{right}}]^T$ is given by

$$\begin{cases} v = \dfrac{r}{2}(w_{\text{right}} + w_{\text{left}}), \\ w = \dfrac{r}{2R}(w_{\text{right}} - w_{\text{left}}). \end{cases} \tag{2}$$

Then, the robot has the following dynamic limitations on velocities:

$$|v| \le v_{\max} \quad \text{and} \quad |w| \le w_{\max}. \tag{3}$$

2.2. Problem setup

The following assumptions are made in this study: (i) the robot has on-board sensors which can detect surrounding objects within a range with a slight margin of error; (ii) the odometry data from the robot has small errors over short distances; (iii) an on-board camera can update at a slower speed relative to local sensors.

It is assumed that the robot knows its initial configuration q_{initial} at the initial time instant t_{initial} and its goal configuration q_{final}. The navigation problem is to design the wheel velocity inputs $[w_{\text{left}}, w_{\text{right}}]^T$ within constraint (3) such that the robot starts at initial configuration, moves collision-free according to a certain performance criterion and arrives in a neighbourhood of the final configuration. To solve this problem, the following choices without loss of generality can be made:*

- The robot's geometric shape is represented by a 2D circle of centre $C = (x, y)$ and of radius R. Its motion

* It is trivial to allow the envelope of either the robot or an obstacle to be represented by union/intersection of several circles. The envelopes could also be polygonal. Mathematically, circular envelopes can be represented by second-order inequalities while polygonal envelopes can be described by first-order linear inequalities.

https://doi.org/10.1017/S0263574708004700 Published online by Cambridge University Press

is controlled but non-holonomic and is represented by the velocity vector $U(t)$. The range of its sensors is also described by a circle centred at C and of radius R_s.
- The ith object, $i = 1, \ldots, N_o$, will be represented by a circle centred at point $O_i = (X_i, Y_i)$ and of radius r_i, denoted by $\mathcal{B}_i(O_i, r_i)$.

3. Motion Planning

Depending on the distance that the robot has to travel, the computation of a complete trajectory from start until finish may be computationally too expensive. Moreover, the environment is partially known and further explored in real time. Therefore, the trajectory has to be computed gradually over time while the mission unfolds. It can be accomplished using an on-line receding horizon planner,[17] in which partial trajectories from an initial state towards the goal are computed by solving an optimal control problem over a limited horizon.

3.1. Receding horizon planner

Contrary to most of the existing trajectory generation modules, the proposed motion planner explicitly takes into account the real time constraint. Indeed, the mobile robot has a limited time to compute its reference optimal trajectory $q_{\text{ref}}(t)$. The time δt allocated to make its decision depends on its perception sensors, its computation delays, etc. The proposed algorithm relaxes the constraint that the final point is reached in the planning horizon. In each receding horizon planning problem, the same planning horizon $T_p \in \mathbb{R}^+$ and update period $T_c \in \mathbb{R}^+$ ($\delta t \leq T_c < T_p$) are used. At each update, denoted τ_k ($k \in \mathbb{N}$),

$$\tau_k = t_{\text{initial}} + kT_c, \tag{4}$$

the robot computes an optimal collision-free trajectory satisfying constraints (1) and (3) using only local information.

To distinguish the different trajectories, the following notations are introduced:

$\widehat{q}(t, \tau_k)$: the predicted trajectory over any interval $[\tau_k, \tau_k + T_p]$
$q_{\text{ref}}(t, \tau_k)$: the optimal planned trajectory over any interval $[\tau_k, \tau_{k+1} = \tau_k + T_c]$,
$q(t)$: the actual trajectory.

The associated control inputs are $\widehat{U}(t, \tau_k)$, $U_{\text{ref}}(t, \tau_k)$ and $U(t)$.

Remark 1 *From the open-loop trajectory and control inputs associated to the planning horizon (i.e., $\widehat{q}(t, \tau_k)$ and $\widehat{U}(t, \tau_k)$), only the part which corresponds to the update horizon is kept (see Fig. 3)*

The motion planning problem is solved in two steps:

- an initialisation step before the robot moves, and
- a step of iterative computations.

Let the following optimal control problem P_{τ_0}, associated with the initialisation step, which consists in determining the optimal predicted control inputs $U_{\text{ref}}(t, \tau_0)$ and the optimal predicted trajectory $q_{\text{ref}}(t, \tau_0)$:

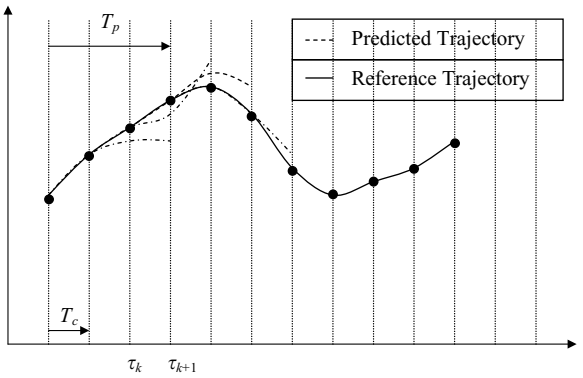


Fig. 3. Planning and update horizons.

$$\min \int_{\tau_0}^{\tau_0+T_p} L(\widehat{q}(t, \tau_0), \widehat{U}(t, \tau_0))\mathrm{d}t + G(\widehat{q}(\tau_0 + T_p, \tau_0), q_{\text{final}}),$$

subject to: $\forall t \in \left[\tau_0, \tau_0 + T_p\right]$,

$$\begin{cases} \dot{\widehat{q}}(t, \tau_0) = f(\widehat{q}(t, \tau_0))\widehat{U}(t, \tau_0), \\ \widehat{q}(\tau_0, \tau_0) = q(\tau_0), \\ \widehat{U}(\tau_0, \tau_0) = U(\tau_0), \\ |\widehat{v}(t, \tau_0)| \leq v_{\max} - \varepsilon_v, \\ |\widehat{w}(t, \tau_0)| \leq w_{\max} - \varepsilon_w, \\ \sqrt{(\widehat{x}(t, \tau_0) - X_i)^2 + (\widehat{y}(t, \tau_0) - Y_i)^2} > R + r_i, \forall \mathcal{B}_i(O_i, r_i) \\ \text{in the range of sensors,} \end{cases}$$

where $L(.)$ indicates a certain performance criterion (time-based criterion, physics-based criterion, geometric-based criterion, etc.) and $G(.)$ represents a terminal penalty function. The latter should be an estimate of the cost to go from the last predicted state $\widehat{q}(\tau_0 + T_p, \tau_0)$ in the planning horizon to the desired final point q_{final}.

Remark 2 *The inclusion of constants $\varepsilon_v \in \mathbb{R}^+$ and $\varepsilon_w \in \mathbb{R}^+$ in the constraints of the motion planning problem guarantees that there is sufficient control authority to track the optimal planned trajectory (see Section 4).*

This process is then repeated during the robot's movement, over the interval $[\tau_0, \tau_1]$, and so on until its reaches a neighbourhood of the goal q_{final}. As such, new detected obstacles can be taken into account in the next iteration. Over any interval $[\tau_k, \tau_{k+1}]$ ($k \in \mathbb{N}$), the following optimal control problem $P_{\tau_{k+1}}$, associated with the $(k+1)$th step, which consists in determining the optimal predicted control inputs $U_{\text{ref}}(t, \tau_{k+1})$ and the optimal predicted trajectory $q_{\text{ref}}(t, \tau_{k+1})$ is solved:

$$\min \int_{\tau_{k+1}}^{\tau_{k+1}+T_p} L(\widehat{q}(t, \tau_{k+1}), \widehat{U}(t, \tau_{k+1}))\, \mathrm{d}t + G(\widehat{q}(\tau_{k+1} + T_p, \tau_{k+1}), q_{\text{final}}), \tag{5}$$

https://doi.org/10.1017/S0263574708004700 Published online by Cambridge University Press

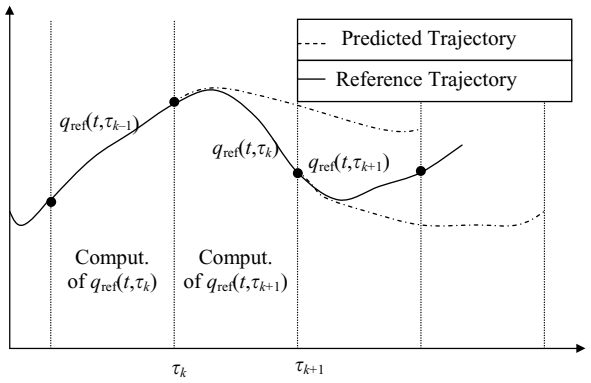


Fig. 4. Implementation of the receding horizon planner.

subject to: $\forall t \in [\tau_{k+1}, \tau_{k+1} + T_p]$,

$$\widehat{\dot{q}}(t, \tau_{k+1}) = f(\widehat{q}(t, \tau_{k+1}))\widehat{U}(t, \tau_{k+1}), \tag{6}$$

$$\widehat{q}(\tau_{k+1}, \tau_{k+1}) = \widehat{q}(\tau_{k+1}, \tau_k), \tag{7}$$

$$\widehat{U}(\tau_{k+1}, \tau_{k+1}) = \widehat{U}(\tau_{k+1}, \tau_k), \tag{8}$$

$$|\widehat{v}(t, \tau_{k+1})| \leq v_{\max} - \varepsilon_v, \tag{9}$$

$$|\widehat{w}(t, \tau_{k+1})| \leq w_{\max} - \varepsilon_w, \tag{10}$$

$$\sqrt{(\widehat{x}(t, \tau_{k+1}) - X_i)^2 + (\widehat{y}(t, \tau_{k+1}) - Y_i)^2} > R + r_i,$$
$$\forall \mathcal{B}_i(O_i, r_i) \text{ in the range of sensors.} \tag{11}$$

Once $P_{\tau_{k+1}}$ is solved, the optimal reference trajectory and control inputs over the interval $[\tau_{k+1}, \tau_{k+2}]$, are stored for use at the next module (i.e., trajectory tracking controller).

Remark 3 *One can note that constraints (7) and (8) on the initial conditions involve the optimal predicted control inputs $\widehat{U}(\tau_{k+1}, \tau_k)$ and trajectory $\widehat{q}(\tau_{k+1}, \tau_k)$ computed in the previous step. Therefore, in the proposed strategy, the receding horizon planner is not used in order to reject external disturbances or inherent discrepancies between the model and the real process, as it is usually done.*[18] *However, it takes the computation time δt into account. Figure 4 gives an overview of the receding horizon planner.*

Remark 4 *A compromise must be found between reactivity and optimality. Indeed, the planning horizon T_p must be sufficiently small in order to have good enough results in terms of optimality for trajectory planning. However, it must be higher than the update horizon T_c in order to guarantee the reactivity and obstacle avoidance for further receding horizon planning problems.*

3.2. Technique for solving receding horizon planning problems

There are three components to the real time resolution of the optimal control problems P_{τ_k} ($k \in \mathbb{N}$): determination of the flat outputs, B-spline parametrisation and constrained feasible sequential quadratic programming.

The method consists in finding outputs such that Eq. (1) is mapped to a lower dimensional output space. It will imply

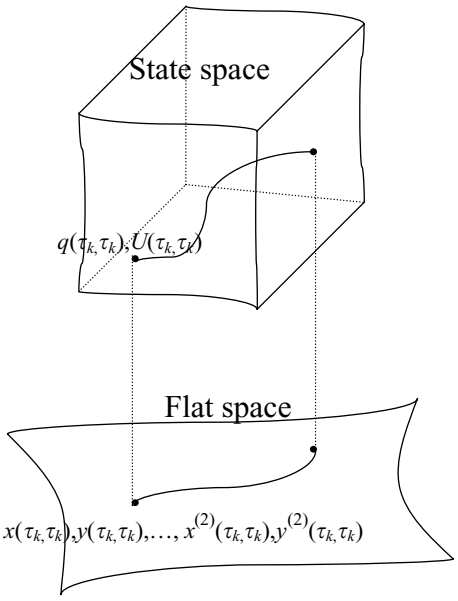


Fig. 5. Flatness and motion planning.

that the problem becomes computationally more efficient to solve. Using the flatness property of system (1) (see ref. [19] for further details about flatness), all system variables can be differentially parameterised by x, y and a finite number of their time derivatives. Indeed, θ, v and w can be expressed as functions of x, y and their first- and second-time derivatives, i.e.,

$$\theta = \arctan\frac{\dot{y}}{\dot{x}}, \quad v = \sqrt{\dot{x}^2 + \dot{y}^2} \quad \text{and} \quad w = \frac{\ddot{y}\dot{x} - \ddot{x}\dot{y}}{\dot{x}^2 + \dot{y}^2}. \tag{12}$$

Once the performance criterion (5) and constraints (6)–(11) are mapped into the flat output space, the optimal predicted trajectory is planned in this space (see Fig. 5).

Then, in order to transform the optimal trajectory generation problem into a parameter optimization one, a piecewise polynomial function, B-spline, is adopted to approximate the reference trajectory. The B-spline functions are chosen as basis functions due to their flexibility and ease of enforcing continuity across breakpoints. B-Spline is the function defined by a series of knots called control knots. In this work, the three-order B-spline basis functions are used to parameterise the trajectory. For problem P_{τ_k}, the time interval $[\tau_k, \tau_k + T_p]$ is divided into n_{knot} equal segments with $n_{\text{knot}} + 4$ knots to be control knots:

$$q_0 = \cdots = q_3 = \tau_k < q_4 < \cdots < q_{n_{\text{knot}}+3} = \tau_k + T_p. \tag{13}$$

The trajectories of the flat outputs are written in terms of finite dimensional B-spline curves as

$$\begin{bmatrix} \widehat{x}(t, \tau_k) \\ \widehat{y}(t, \tau_k) \end{bmatrix} = \sum_{j=1}^{3+n_{\text{knot}}} C_j B_{j,3}(t), \tag{14}$$

https://doi.org/10.1017/S0263574708004700 Published online by Cambridge University Press

where $C_j \in \mathbb{R}^2$ are the coefficients of the third-order B-spline and $B_{j,3}$ is the B-spline basis function computed recursively as follows:

$$B_{j,0}(t) = \begin{cases} 1 & \text{if } q_j \le t < q_{j+1}, \\ 0 & \text{otherwise .} \end{cases}$$
$$\forall d \in \{1, 2, 3\}, \tag{15}$$
$$B_{j,d}(t) = \frac{t - q_j}{q_{j+d+2} - q_j} B_{j,d-1}(t) + \frac{q_{j+d+1} - t}{q_{j+d+1} - q_{j+1}} B_{j+1,d-1}(t).$$

Finally, the time domain is truncated into smaller intervals by quadratic laws. The optimal coefficients C_j are numerically found using the constrained feasible sequential quadratic optimization algorithm.[20] See ref. [21] for a detailed analysis of the efficiency of this approach. Eventually, the open-loop control inputs are deduced using Eq. (12).

4. Trajectory Tracking Controller

The task, for this module, is to design the wheel velocities such that the robot tracks the reference trajectory generated in the previous section.

4.1. Formulation of the tracking problem

The reference trajectory, (x_r, y_r, θ_r), generated by the motion planning algorithm fulfills the differential equation

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{bmatrix} = \begin{bmatrix} \cos\theta_r & 0 \\ \sin\theta_r & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_r \\ w_r \end{bmatrix}, \tag{16}$$

where the desired velocities v_r and w_r satisfy

$$|v_r| \le v_{\max} - \varepsilon_v \quad \text{and} \quad |w_r| \le w_{\max} - \varepsilon_w.$$

By directly applying v_r and w_r, the robot does not follow the reference trajectory with good accuracy. It is obvious that the real control inputs v and w rely on the state measurements x, y and θ. Due to measurement noise and modelling uncertainties, there are input uncertainties for v and w. That is to say, the actual equation of the robot trajectory fulfills the following uncertain differential equation:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v + \delta_v \\ w + \delta_w \end{bmatrix}, \tag{17}$$

where δv and δw represent the uncertainties.

Control inputs v and w must be designed such that system (17) follows reference (16) in spite of the perturbations. In fact, the goal is to asymptotically stabilize the tracking errors $e_x = x_r - x$, $e_y = y_r - y$ and $e_\theta = \theta_r - \theta$ to zero while respecting the saturation constraints (3).

Transforming the tracking errors expressed in the inertial frame to the robot frame (see ref. [22]), the error coordinates can be denoted as

$$\begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} e_x \\ e_y \\ e_\theta \end{bmatrix}.$$

Accordingly, the tracking error dynamics is represented by

$$\dot{e} = f_1(e, t) + f_2(e, t)(U + \delta), \tag{18}$$

where

$$\begin{cases} e & = [e_1, e_2, e_3]^T, \\ U & = [v, w]^T, \\ \delta & = [\delta_v, \delta_w]^T, \\ f_1(e, t) & = \begin{bmatrix} v_r \cos e_3 \\ v_r \sin e_3 \\ w_r \end{bmatrix}, \\ f_2(e, t) & = \begin{bmatrix} -1 & e_2 \\ 0 & -e_1 \\ 0 & -1 \end{bmatrix}. \end{cases}$$

It should be pointed out that such a system cannot be stabilized by continuously differentiable, time-invariant state feedback control laws. In this paper, integral sliding mode control is combined with time-varying state feedback in order to further robustify against perturbations.

4.2. Proposed methodology

The basic idea is to use an integral sliding mode controller to reject the matched perturbation (i.e., perturbations that enter the state equation at the same point as the control inputs). The integral sliding mode control algorithm is designed in two steps (see refs. [23, 24]):

1. the selection of a suitable integral sliding variable s such that, while sliding, the control objective is fulfilled, and
2. the design of corresponding control inputs U constraining the system trajectories to evolve on the sliding surface from the initial time instant.

For system (18), the control law is defined as follows:

$$U(q, t) = U_0(q, t) + U_1(q, t). \tag{19}$$

The nominal control $U_0(q, t)$ is responsible for the performance of the nominal system. $U_1(q, t)$ is a discontinuous control action that rejects the perturbations by ensuring the sliding motion.

4.2.1. Integral sliding mode controller. Let us define the sliding variable $s(q, t) = [s_1(q, t), s_2(q, t)]^T \in \mathbb{R}^2$ as

$$s(q, t) = P\left[e(t) - e(t_{\text{initial}}) - \int_{t_{\text{initial}}}^{t} (f_1(e, \nu) + f_2(e, \nu) U_0(q, \nu))\, \mathrm{d}\nu\right], \tag{20}$$

https://doi.org/10.1017/S0263574708004700 Published online by Cambridge University Press

where matrix $P \in \mathbb{R}^{2\times 3}$ is such that $Pf_2(e,t)$ is invertible for all $t \in \mathbb{R}^+$. Setting $P = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}$, the above condition is fulfilled. One can note that, at the initial time instant $t = t_{\text{initial}}$, the sliding variable satisfies $s(q,t) = 0$, such that the controlled system always starts on the sliding surface $\{s(q,t) = 0\}$.

Remark 5 *To simplify notation, we will omit some of the functions arguments from now onwards.*

Based on the following Lyapunov function candidate, $V = \frac{1}{2}s^T s$, the discontinuous control term can be determined such that $\dot{V} < 0$, guaranteeing the attractivity of the sliding surface. Indeed, one has

$$\begin{aligned}\dot{V} &= s^T(P\dot{e} - P(f_1 + f_2U_0)) \\ &= s^T(P(f_1 + f_2(U + \delta)) - P(f_1 + f_2U_0)) \\ &= s^T(Pf_2(U_1 + \delta)) \\ &= ([Pf_2]^T s)^T(U_1 + \delta) \\ &= [s_1 - e_2 s_1 + s_2](U_1 + \delta) < 0.\end{aligned}$$

The above condition holds if

$$U_1 = \begin{bmatrix} -M_1 \text{sign}(s_1) \\ -M_2 \text{sign}(-e_2 s_1 + s_2) \end{bmatrix}, \tag{21}$$

where M_1 and M_2 are gains high enough to enforce the sliding motion.

The trajectory evolves on $\{s = 0\}$ from $t = t_{\text{initial}}$ and remains there in spite of the perturbations. To determine the motion equations on the sliding surface, the equivalent control method[25] is used. The time derivative of the sliding variable is

$$\begin{aligned}\dot{s} &= P(\dot{e} - f_1 - f_2U_0) \\ &= Pf_2(U_1 + \delta).\end{aligned}$$

The equivalent control is obtained by solving equation $\dot{s} = 0$ for U_1

$$U_{1\text{eq}} = -\delta. \tag{22}$$

By substituting $U_{1\text{eq}}$ for U_1 in (18), one obtains the sliding dynamics

$$\dot{e} = f_1(e) + f_2(e)U_0. \tag{23}$$

Remark 6 *From Eq. (23), several conclusions can be drawn. First, the sliding motion is insensitive with respect to the matched perturbation: it has been successfully rejected. Second, with respect to the conventional sliding mode control, we have gained some extra degrees of freedom. Indeed, U_0 can be used in order to stabilize the nominal system and to deal with the unmatched perturbation.*

4.2.2. Time-varying state feedback controller. The second part of the controller design is to find a saturated control law U_0 such that the sliding dynamics (23) is globally asymptotically stable. The chosen control law U_0 has been developed by Jiang et al.[26] The motivation for such a choice is that this design takes the actuator bounds into account. It is described by

$$U_0 = \begin{bmatrix} v_0 = v_r \cos e_3 + \lambda_3 \tanh e_1 \\ w_0 = w_r + \dfrac{\lambda_1 v_r e_2}{1 + e_1^2 + e_2^2}\dfrac{\sin e_3}{e_3} + \lambda_2 \tanh e_3 \end{bmatrix}. \tag{24}$$

Note that the positive parameters λ_1, λ_2 and λ_3 can be designed such that the bounds of the controls are met for our controllers. Indeed, it can be seen that

$$|v_0| \le v_{\max} - \varepsilon_v + \lambda_3, \quad |w_0| \le w_{\max} - \varepsilon_w + \frac{\lambda_1(v_{\max} - \varepsilon_v)}{2} + \lambda_2.$$

Remark 7 *The control gains can be designed such that the bounds on the control inputs are satisfied. In order to design these constants, a compromise must be found between the optimality, the performance and the robustness of the control law with respect to perturbations.*

From (2), it is straightforward to obtain the wheel velocities

$$\begin{cases} w_{\text{left}} = \frac{v - Rw}{r}, \\ w_{\text{right}} = \frac{v + Rw}{r}. \end{cases}$$

5. Experiments

5.1. Experimental setup

The proposed motion planning and control algorithms are implemented on the mobile robot Pekee manufactured at Wany Robotics company (see Fig. 6). An overview of the experimental setup is shown in Fig. 7. An Intel 486 micro-processor running at 75 MHz hosts the integral sliding mode controller written in C language. The robot operates under Linux real time and its software integrates sensor and communication data. It communicates through wireless Ethernet capable of transmitting data up to 3 Mb/s. One miniature colour vision camera C-Cam8 is on the robot.

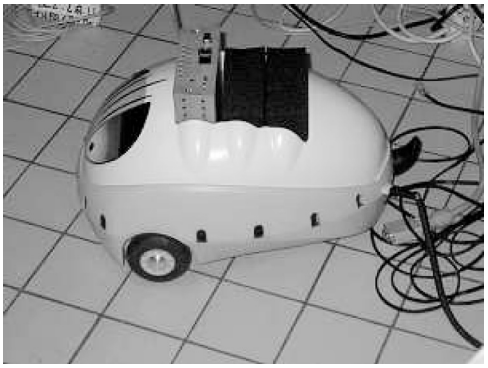


Fig. 6. Pekee mobile robot.

https://doi.org/10.1017/S0263574708004700 Published online by Cambridge University Press

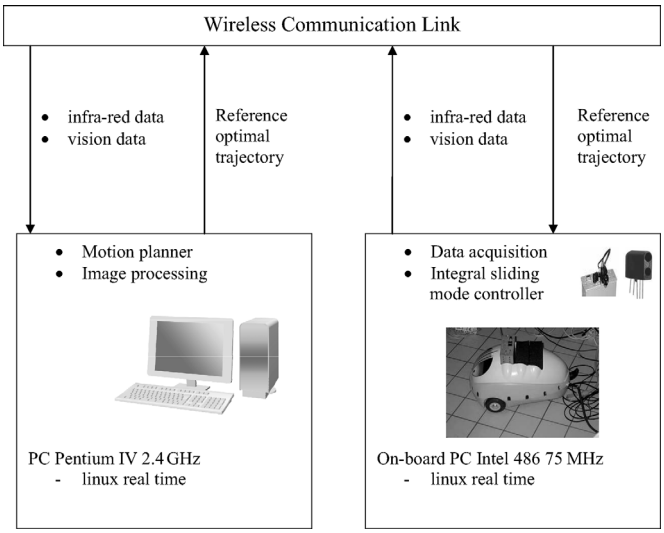


Fig. 7. Overview of the experimental setup.



Fig. 8. Examples of cluttered and partially known environments.

A C programme accesses the streaming data coming into the frame grabber from the camera and stores the data in a 320×240 image file. The robot is also equipped with 15 infra-red sensors used for local identification of the environment and two encoders.

The vehicle wheelbase is taken as $R = 0.3$ m with $v_{\max} = 0.7$ m/s, $w_{\max} = 3$ rad/s and the sampling period is 100 ms. The computation time δt including the image processing and the motion planning algorithm is about 2 min on the on-board 75 MHz PC. In order to decrease the computation delay, the socket protocol communication and wireless communication links were used. The vision data are sent to a Pentium IV 2.4 GHz PC for image processing and for the generation of the optimal trajectory. This protocol enables to reduce δt to less than 0.4 s.

5.2. *Experimental results*

Experiments were made in different cluttered and partially known environments.*

In these experiments, obstructed areas are created with circular obstacles in the workspace (see Fig. 8). Some of them are initially out of view from the robot on-board sensors and are discovered during the robot movement. The performance criterion is the length of the travelled distance.

For the motion planner, the chosen parameters are described in Table I. The planning horizon T_p is chosen according to the range of sensors (denoted R_s) and the maximum linear velocity $v_{\max}$ (i.e., $T_p < \frac{R_s}{v_{\max}}$). The update horizon T_c must be upper than the computation time δt. Parameter n_{knot} is tuned in order to achieve a good compromise between computation time (small value of n_{knot}) and optimality (high value of n_{knot}). Since one of our objectives is to accurately track the planned trajectory, constants ε_v and ε_w are selected in order to have sufficient control authority to reject the external disturbances and inherent discrepancies between the model and the real process. After some analysis of measurements noise and other uncertainties, ε_v and ε_w are set to 0.2 m/s and 1 rad/s, respectively.

Table I. Parameters of the motion planner.

T_p	2 s
T_c	0.5 s
n_{knot}	6
ε_v	0.2 m/s
ε_w	1 rad/s

Table II. Parameters of the tracking algorithm.

λ_1	0.5
λ_2	1
λ_3	0.5
M_1	0.15 m/s
M_2	0.75 rad/s

For the tracking algorithm, the chosen parameters are given in Table II. The switching gains M_1 and M_2 are selected according to the estimation of input uncertainties for v and w (i.e., $M_1 \geq \delta_v$ and $M_2 \geq \delta_w$). Eventually, parameters λ_i ($i = 1, 2, 3$) must be positive. If there are some initial localization uncertainties, the convergence rate of the tracking errors directly depends on the value of these constants. For comparison purpose, gains M_1 and M_2 needed to enforce the sliding mode are the same in all the experiments. The discontinuous controls are approximated by

$$U_1 = \begin{bmatrix} -M_1 \dfrac{s_1}{|s_1| + 0.0001} \\ -M_2 \dfrac{-e_2 s_1 + s_2}{|-e_2 s_1 + s_2| + 0.0001} \end{bmatrix}$$

Figures 9(a) and 10(a) depict the executed trajectories in unknown environments. First, the robot visualizes the scene and applies the image processing. The nearest obstacles in view from its on-board sensors are detected. In order to take the size of the robot into account, the radius of these obstacles is increased by 0.3 m (dotted lines around obstacles). According to the detected obstacles, a collision-free trajectory is planned. Then, the integral sliding mode controller enables to track the desired trajectory in spite of uncertainties and errors. During the execution, the robot plans, in real time, its next optimal collision-free trajectory by taking into account new information from its infra-red and camera sensors. The effectiveness, perfect performance of obstacle avoidance, real time and high robustness properties are demonstrated by these experimental results.

* We invite the reader to see more experimental results on http://www.isen.fr/~sst_lille/fichiers/Icra.wmv.

https://doi.org/10.1017/S0263574708004700 Published online by Cambridge University Press

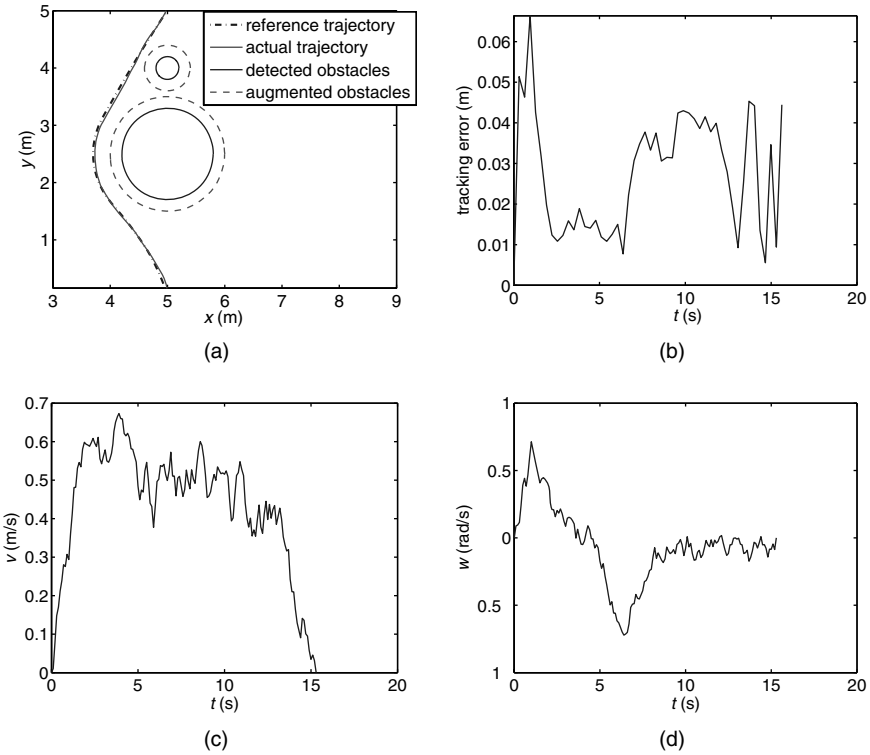


Fig. 9. Experimental results using the integral sliding mode and time-varying state feedback controllers.

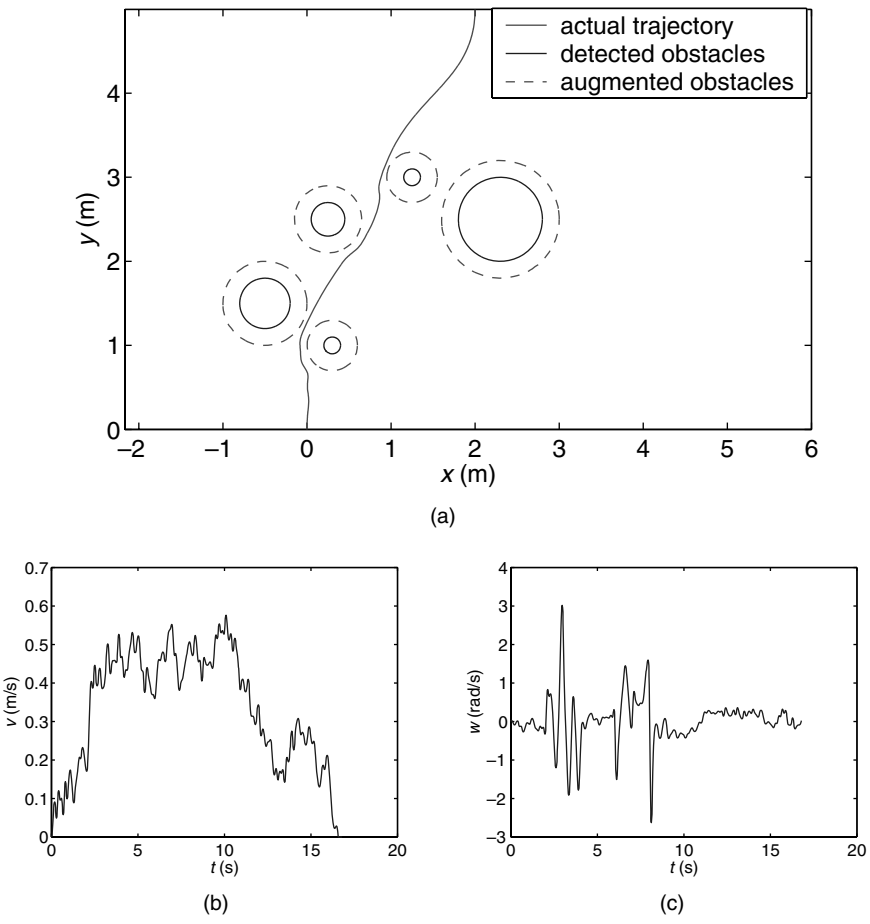


Fig. 10. Experimental results using the integral sliding mode and time-varying state feedback controllers in a more complex map.

https://doi.org/10.1017/S0263574708004700 Published online by Cambridge University Press

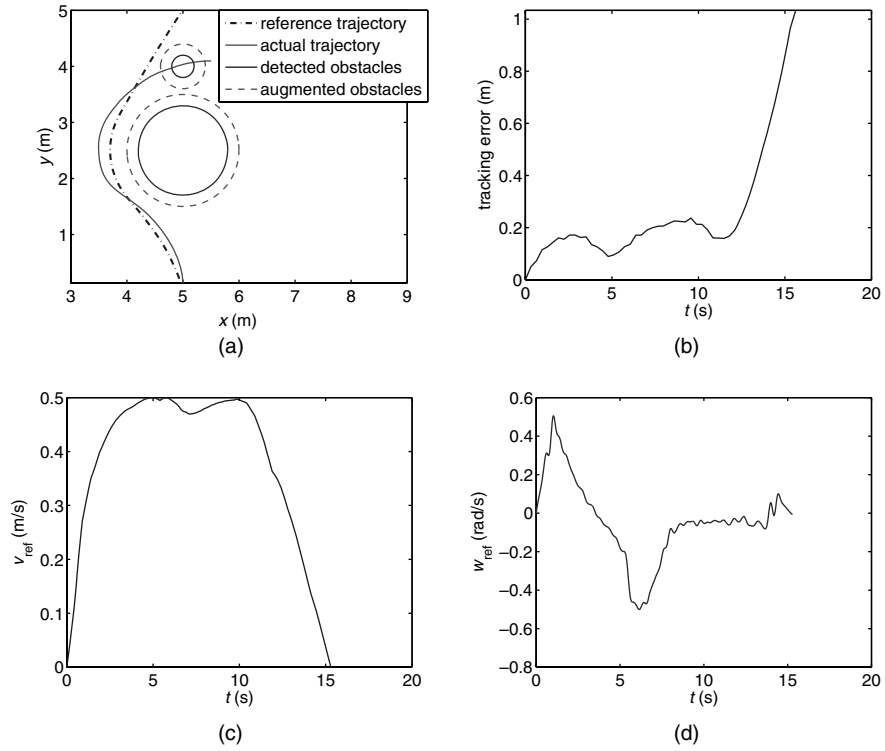


Fig. 11. Experimental results without using the trajectory tracking module.

Figures 9, 11 and 12 highlight the performance for the proposed tracking module. The first experiment is made without using the feedback controller (Fig. 11). By directly applying the open-loop control inputs, the robot does not follow its planned trajectory due to measurement noise and modelling uncertainties. One can see in Fig. 11(b) that the distance between the actual and planned trajectories, i.e., $\sqrt{(x - x_r)^2 + (y - y_r)^2}$, diverges. The second one is made using only the time-varying state feedback controller (Fig. 12). The third experiment is done using the combination 'integral sliding mode controller plus time-varying state feedback controller' (Fig. 9). One can note that using only

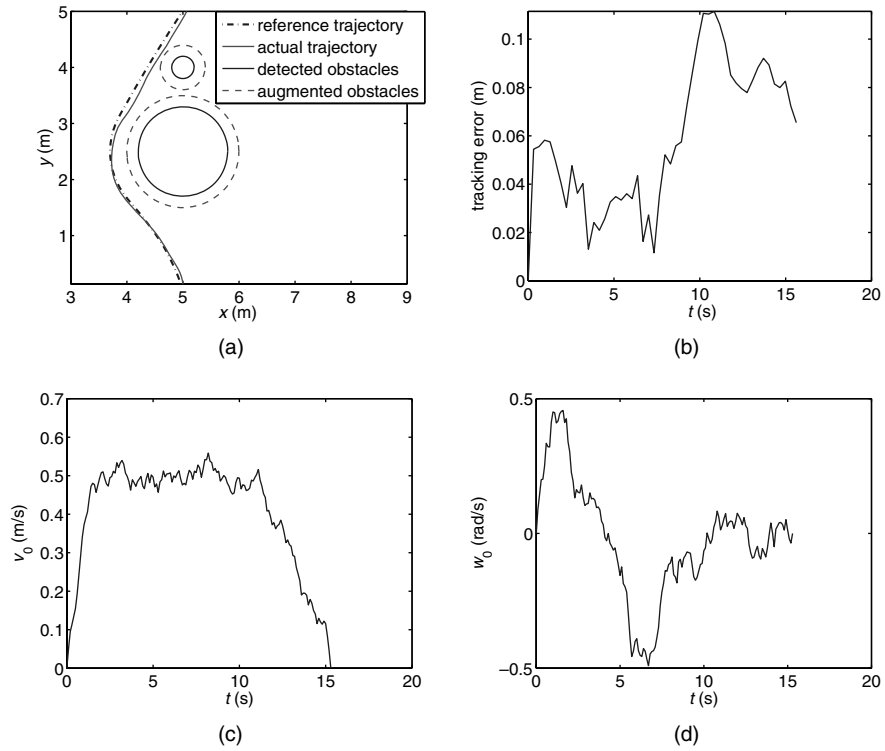


Fig. 12. Experimental results using the time-varying state feedback controller.

https://doi.org/10.1017/S0263574708004700 Published online by Cambridge University Press

the time-varying state feedback controller, the robustness performance is not good enough. The use of an integral sliding mode controller enables to improve the accuracy of the tracking motion.

6. Conclusion

An architecture for real time navigation of an autonomous mobile robot evolving in an uncertain environment with obstacles is proposed. It provides the following practical advantages:

- First, it enables to take into account the dynamic limitations on velocities.
- Then, receding horizon planner is a viable method for real-time trajectory generation. Depending on computing resources, the use of flatness, B-spline parametrisation and constrained feasible sequential quadratic programming can take less than 1 s to compute an optimal collision-free trajectory.
- Furthermore, the combination of integral sliding mode control with other methods like time-varying state feedback improves the robustness properties of the closed-loop system. Therefore, high precision trajectory tracking is achieved in spite of uncertainties and modelling errors.

Experimental results show the effectiveness of our practical scheme (real time, high robustness properties and good performance for obstacle avoidance).

Acknowledgements

This work was supported by the JSPS Postdoctoral Fellowship, Region Nord Pas-de-calais and the FEDER (European Funds of Regional Development) under the Interreg **ACOS**, ARCIR Robocoop and the AUTORIS-TAT T31 project.

References

1. J.-P. Laumond, *Robot Motion Planning and Control* (Springer, New York, 1998).
2. T. Statheros, M. Defoort, S. Khola, K. McDonald-Maier, W. Howells, A. Kokosy, J. Palos, W. Perruquetti and T. Floquet, "Automated Control and Guidance System (ACOS): An Overview," *International Conference on Recent Advances in Soft Computing*, Canterbury, UK (2006).
3. M. Salichs and L. Moreno, "Navigation of mobile robots: Open questions," *Robotica* **18**, 227–234 (2000).
4. M. Galicki, "Time-optimal controls of kinematically redundant manipulators with geometric constraints," *IEEE Trans. Rob. Automat.* **16**(1), 89–93 (2000).
5. A. Bloch, *Nonholonomic Mechanics and Control* (Springer Verlag, New York, 2003).
6. R. Brockett, "Asymptotic Stability and Feedback Stabilization," **In**: *Differential Geometric Control Theory* (R. Brockett, R. Millman and H. Sussmann, eds.) (Birkhauser, Boston, MA, 1983) pp. 181–195.
7. J. Pomet, "Explicit design of time-varying stabilizing control laws for a class of controllable systems without drift," *Syst. Cont. Let.* **18**(2), 147–158 (1992).
8. C. Samson, "Control of chained systems: Application to path following and time-varying point-stabilization of mobile robots," *IEEE Trans. Automat. Control* **40**, 64–77 (1995).
9. R. Murray and S. Sastry, "Nonholonomic motion planning: Steering using sinusoids," *IEEE Trans. Automat. Control* **38**(5), 700–716 (1993).
10. J. P. Hespanha and A. S. Morse, "Stabilization of non-holonomic integrators via logic-based switching," *Automatica* **35**(3), 385–393 (1999).
11. Z. P. Jiang and H. Nijmeijer, "Tracking control of mobile robots: A case study in backsteeping," *Automatica* **33**(7), 1393–1399 (1997).
12. W.-J. Evers and H. Nijmeijer, "Practical Stabilization of a Mobile Robot Using Saturated Control," *IEEE International Conference on Decision and Control*, San Diego, CA (2006) pp. 2394–2399.
13. T. Floquet, J. P. Barbot and W. Perruquetti, "Higher-order sliding mode stabilization for a class of nonholonomic perturbed systems," *Automatica* **39**(6), 1077–1083 (2003).
14. M. Defoort, J. Palos, T. Floquet, A. Kokosy and W. Perruquetti, "Practical Stabilization and Tracking of a Wheeled Mobile Robot with Integral Sliding Mode Controller," *IEEE International Conference on Decision and Control*, New Orleans, LA (2007) pp. 1999–2004.
15. S. Monaco and D. Normand-Cyrot, "An Introduction to Motion Planning Under Multirate Digital Control," *IEEE International Conference on Decision and Control*, Tuckson, AZ (1992) pp. 1780–1785.
16. C. de Wit and O. Sordalen, "Exponential stabilization of mobile robots with nonholonomic constraints," *IEEE Trans. Automat. Control* **37**(11), 1791–1797 (1992).
17. D. Mayne, J. Rawlings, C. Rao and P. Scokaert, "Constrained model predicitive control: Stability and optimality," *Automatica* **36**(6), 789–814 (2000).
18. F. A. Cuzzolaa, J. C. Geromel and M. Morari, "An improved approach for constrained robust model predictive control," *Automatica* **38**(7), 1183–1189 (2002).
19. M. Fliess, J. Lévine, Ph. Martin and P. Rouchon, "Flatness and defect of nonlinear systems: Introductory theory and examples," *Int. J. Control* **61**(6), 1327–1361 (1995).
20. C. Lawrance, J. Zhou and A. Tits, *User's guide for CFSQP Version 2.5* (Institute for Systems Research, University of Maryland, College Park, 1997).
21. M. B. Milam, Real Time Optimal Trajectory Generation for Constrained Dynamical Systems *Dissertation* (California Institute of Technology, 2003).
22. Y. Kanayama, Y. Kimura, F. Miyazaki and T. Noguchi, "A Stable Tracking Control Method for an Autonomous Mobile Robot," *IEEE International Conference on Robotics and Automation*, Cincinnati, OH (1990) pp. 384–389.
23. V. Utkin and J. Shi, "Integral Sliding Mode in Systems Operating Under Uncertainty Conditions," *IEEE International Conference on Decision and Control*, Kobe, Japan (1996) pp. 4591–4596.
24. M. Defoort, T. Floquet, A. Kokosy and W. Perruquetti, "Integral sliding mode control for trajectory tracking of a unicycle type mobile robot," *Integr. Comput. Aided Eng.* **13**(3), 277–288 (2006).
25. V. Utkin, J. Guldner and J. Shi, *Sliding Modes Control in Electromechanical Systems* (Taylor and Francis, London, 1999).
26. Z. P. Jiang, E. Lefeber and H. Nijmeijer, "Saturated stabilization and track control of a nonholonomic mobile robot," *Syst. Cont. Let.* **42**(5), 327–332 (2001).

https://doi.org/10.1017/S0263574708004700 Published online by Cambridge University Press