# Performance-Based Reactive Navigation for Nonholonomic Mobile Robots

Michael Defoort, Jorge Palos, Annemarie Kökösy, Thierry Floquet, Wilfrid Perruquetti

# Performance–Based Reactive Navigation for Nonholonomic Mobile Robots*

Michael Defoort[1†], Jorge Palos[2], Annemarie Kokosy[2],
Thierry Floquet[1] and Wilfrid Perruquetti[1]

[1]LAGIS UMR CNRS 8146, Ecole Centrale de Lille,
BP 48, Cité Scientifique, 59651 Villeneuve-d'Ascq, FRANCE
[2]ISEN, 41 bvd Vauban, 59 046 Lille Cedex, FRANCE

**Abstract**

This paper presents an architecture for the navigation of an autonomous mobile robot evolving in environments with obstacles. Instead of addressing motion planning and control in different contexts, these issues are described in connected modules with performance requirement considerations. The planning problem is formulated as a constrained receding horizon planning problem and is solved in real time with an efficient computational method that combines nonlinear control theory, B-spline basis function and nonlinear programming. An integral sliding mode controller is used for trajectory tracking. 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:** Nonholonomic 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 toward solving mobile robot navigation in different applications in indoor and outdoor environments (see [1, 2] and the survey paper [3]). For

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

†Corresponding author. E-mail: michael.defoort@ec-lille.fr

some navigation tasks like planetary exploration, robots are required to travel long distances within constrained resources (energy, time…). In such cases, efficient motion planning and control algorithms are needed in order to achieve the goal while meeting certain performance issue, such as geometric-based or time-based criteria.

Although motion planning and control 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 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 performance 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 will not be applicable to high speed traveling. In this research, we bridge the gap between trajectory planning and motion control.

Many theoretically challenging properties stem from the so-called nonholonomic constraints imposed by the rolling wheels. A survey of nonholonomic control problems can be found in [5]. Obstacles to the tracking of nonholonomic 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] or discontinuous controls [12]. However, most of these methods do not provide both fast convergence and good robustness properties. Most of the control laws ensuring exponential or finite time convergence [13] are known to be non-robust under disturbances or modeling 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 nonholonomic 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 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 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 nonholonomic model is described. Motion planning is discussed in Section 3. Trajectory tracking strategy is presented in Section 4. Finally, in Section

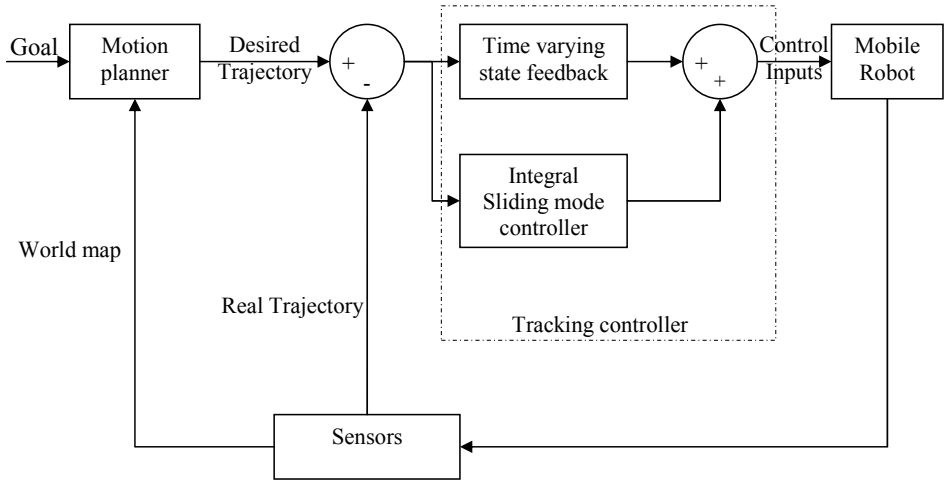



Figure 1: Block diagram for navigation of a nonholonomic mobile robot with a motion planner and a sliding mode controller.

5, we present the integration of the different modules and the experimental results on a mobile robot.

# 2 Problem statement

## 2.1 Modeling of mobile robot

The mobile robot, shown in Fig. 2, is of unicycle-type. The robot body is of symmetric shape and the centre of mass 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 centrered 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, we assume that the motion of the passive wheel can be ignored in 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 $\theta$ is its orientation in the global frame.

In this paper, kinematics of wheeled-mobile robot are shown under the nonholonomic constraints (see [14] for details). The pure rolling and nonslipping nonholonomic conditions are described by:

$$A^T(q)\dot{q} = 0 \quad \text{with} \quad A^T(q) = \begin{bmatrix} -\sin\theta & \cos\theta & 0 \end{bmatrix}.$$

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 inputs, $v$ and $w$ are the linear and angular velocities, respectively. The relationship between $[v, w]^T$ and the left and right velocities $\left[w_{left}, w_{right}\right]^T$ is described by:

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

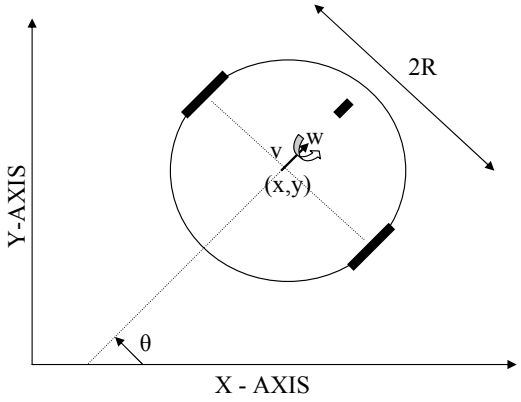



Figure 2: Unicycle-type mobile robot

At last, the robot has the following dynamic limitations on velocities:

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

## 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 small 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 decide wheel velocity inputs $\left[w_{left}, w_{right}\right]^T$ within constraint (3) such that the robot starts at initial configuration, moves collision-free according to a certain performance criteria

and arrives in a neighbourhood of the final configuration. To solve this problem, one can make the following choices without loss of any generality[1]:

- The robot's geometric shape is represented by a 2-D circle of centre $C = (x,y)$ and of radius $R$. Its motion is controlled but nonholonomic and is represented by the velocity vector $U(t)$. The range of its sensors is also described by a circle centrered at $C$ and of radius $R_s$.
- The $i^{textth}$ object, $i = 1,\ldots,N_o$, will be represented by a circle centrered at point $O_i = (X_i, Y_i)$ and of radius $r_i$, denoted by $\mathscr{B}_i(O_i, r_i)$.

# 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 [15], in which partial trajectories from an initial state toward 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 $q_{ref}(t)$. The time $\delta t$ 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 $\tau_k$ ($k \in \mathbb{N}$),

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

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

To distinguish the different trajectories, we introduce the following notations:

| | |
|---|---|
| $\widehat{q}(t,\tau_k)$ : | the predicted trajectory over any interval $[\tau_k, \tau_k + T_p]$, |
| $q_{ref}(t,\tau_k)$ : | 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 $\widehat{U}(t,\tau_k)$, $U_{ref}(t,\tau_k)$ and $U(t)$.

[1] 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.

**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).*

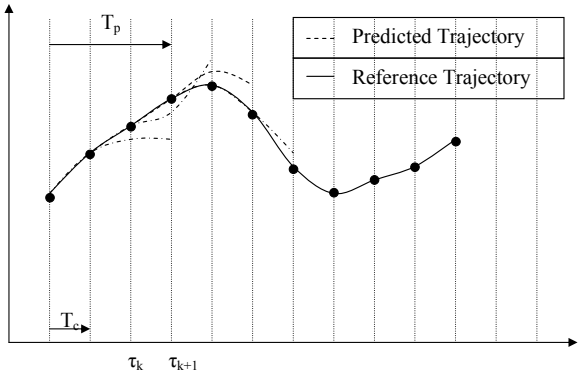



Figure 3: Planning and update horizons

The motion planning problem is solved in two steps:

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

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

$$\min \int_{\tau_0}^{\tau_0+T_p} L(\widehat{q}(t,\tau_0),\widehat{u}(t,\tau_0))dt + G(\widehat{q}(\tau_0+T_p,\tau_0),q_{final}),$$

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

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

where $L(.)$ indicates a certain performance criteria (time-based criteria, physics-based criteria, geometric-based criteria, 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 $(k+1)^{\text{th}}$ step, which consists in determining the optimal predicted control inputs $u_{ref}(t, \tau_{k+1})$ and the optimal predicted trajectory $q_{ref}(t, \tau_{k+1})$ is solved:

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

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

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

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

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

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

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

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

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 need 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 [16]. However, it takes into account the computation time $\delta t$. Fig. 4 gives an overview of the receding horizon planner.*

**Remark 4** *A compromise must be done between reactivity and optimality. Indeed, the planning horizon $T_p$ must be sufficiently small in order to have good enough results in term 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 next 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 $P_{\tau_k}$ ($k \in \mathbb{N}$): determination of the flat outputs, B-spline parametrization and constrained feasible sequential quadratic programming.

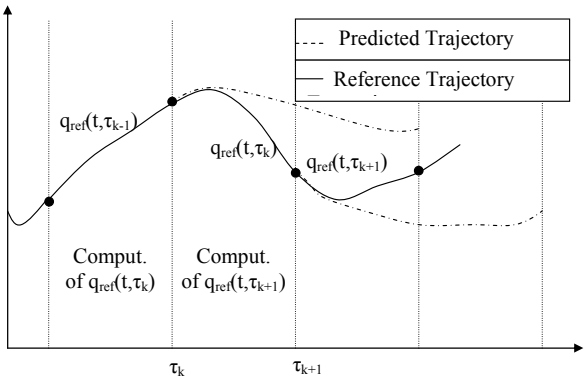



Figure 4: Implementation of the receding horizon planner

The key approach is to determine outputs such that equation (1) is mapped to a lower dimensional output space. It will imply that the problem becomes computionally more efficient to solve. Using the flatness property of system (1) (see [17] for further details about flatness), all system variables can be differentially parameterized by $x$, $y$ and a finite number of their time derivatives. Indeed, $\theta$, $v$ and $w$ can be expressed by $x$, $y$ 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 criteria (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 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 our study, the three-order B-spline basis functions are used to parameterize the trajectory. For problem $P_{\tau_k}$, the time interval $[\tau_k, \tau_k + T_p]$ is divided into $n_{knot}$ equal segments with $n_{knot}+4$ knots to be control knots:

$$nod_0 = \ldots = nod_3 = \tau_k < nod_4 < \ldots < nod_{n_{knot}+3} = \tau_k + T_p \tag{13}$$

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

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

where $C_j \in \mathbb{R}^2$ are the coefficients of the third-order B-spline and $B_{j,3}$ is the B-spline

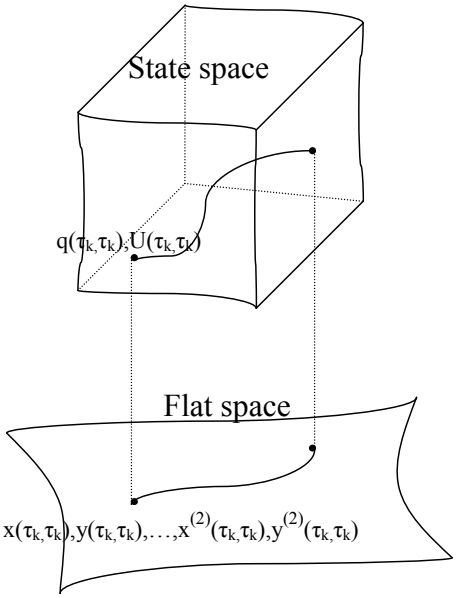



Figure 5: Flatness and motion planning

basis function computed recursively as follows:

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

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 [18]. See [19] for a detailed analysis of the efficiency of this approach. To finish, the open-loop control inputs are deduced using equation (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 one.

## 4.1 Formulation of the tracking problem

The reference trajectory $(x_r, y_r, \theta_r)$, 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:

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

By directly applying $v_r$ and $w_r$, the robot does not follow the reference trajectory with a good accuracy. It is obvious that the real control inputs $v$ and $w$ rely on the state measurements $x$, $y$ and $\theta$. Due to measurement noise and modeling 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 $\delta v$ and $\delta w$ 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, 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, we combine integral sliding mode control 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 [20]:

1. the selection of a suitable integral sliding variable $s$ such that, while sliding, the control objective is fulfilled,
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 reject 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:

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

where matrix $P \in \mathbb{R}^{2\times 3}$ is such that $Pf_2(e,t)$ is invertible for all $t \in \mathbb{R}^+$. Making $P = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}$, the above condition is fulfilled. One can note that, at the initial time instant $t = t_{initial}$, 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 on.*

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. One can obtain:

$$\begin{aligned}\dot{V} &= s^T\left(P\dot{e} - P(f_1 + f_2U_0)\right)\\ &= s^T\left(P(f_1 + f_2(U+\delta)) - P(f_1 + f_2U_0)\right)\\ &= s^T\left(Pf_2(U_1+\delta)\right)\\ &= \left([Pf_2]^T s\right)^T (U_1+\delta)\\ &= \begin{bmatrix} s_1 & -e_2s_1 + s_2\end{bmatrix}(U_1+\delta) < 0\end{aligned}$$

The above condition holds if:

$$U_1 = \begin{bmatrix} -M_1\,\mathrm{sign}\,(s_1) \\ -M_2\,\mathrm{sign}\,(-e_2s_1 + s_2)\end{bmatrix} \tag{21}$$

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

The trajectory evolves on $\{s = 0\}$ from $t = t_{initial}$ and remains there in spite of the perturbations. To determine the motion equations on the sliding surface, the equivalent control method [21] 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 the equation $s = 0$ for $U_1$:

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

By substituting $U_{1eq}$ for $U_1$ in (18), one can obtain the sliding dynamics:

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

**Remark 6** *From equation (23), several conclusions can be drawn. Firstly, the sliding dynamics do not contain the matched perturbation: it has been successfully rejected. Secondly, 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 in [22]. The motivation for such a choice is that this design takes into account the actuator bounds. It is described by:

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

Note that the positive parameters $\lambda_1$, $\lambda_2$ and $\lambda_3$ can be designed such that the bounds of the controls are met for our controllers. Indeed, it can be seen that:

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

**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 with respect to perturbations.*

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

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

# 5 Experimental results

## 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 $75MHz$ hosts the integral sliding mode controller written in C. 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 $3Mb/s$. One miniature color vision camera C-Cam8 is mounted on the robot. A C program accesses the streaming data coming into the frame grabber from the camera and stores the data in a $320 \times 240$ image file. The robot is also equipped with 15 infra-red sensors used for local identification of the environment and two encoders.

Figure 6: Pekee mobile robot

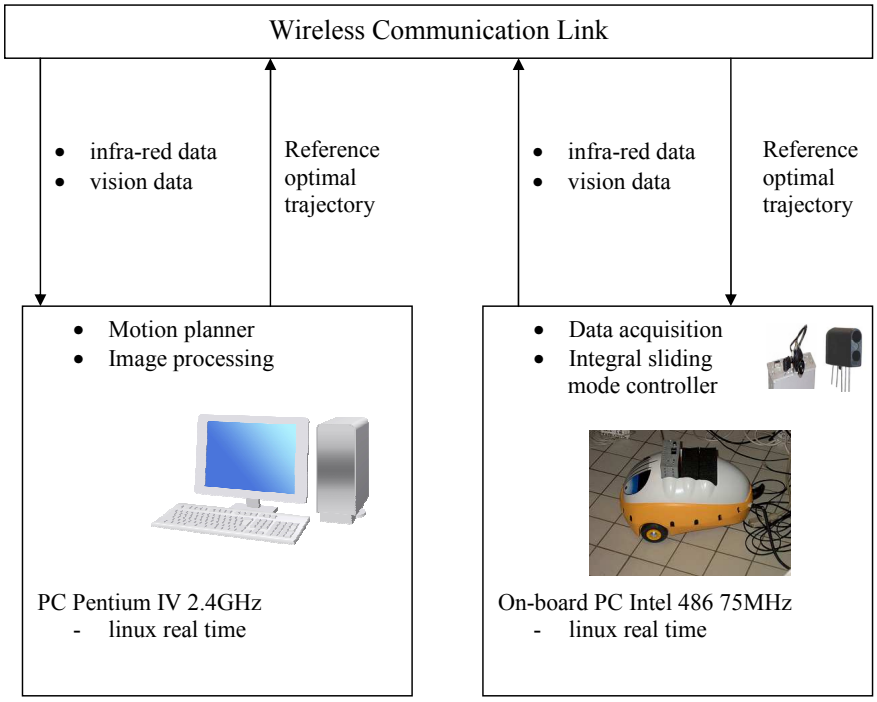



Figure 7: Overview of the experimental setup

The vehicle's wheelbase is taken to $R = 0.3m$ with $v_{max} = 0.8m/s$, $w_{max} = 5rad/s$ and the sampling period is $100ms$. The computation time $\delta t$ including the image processing and the motion planning algorithm is about two minutes on the on-board $75MHz$ PC. In order to decrease the computation delay, we used the socket protocol communication and wireless communication link. The vision data are sent to a Pentium IV $2.4GHz$ PC for image processing and for the generation of the optimal trajectory. This protocol enables to reduce $\delta t$ to less than $0.2s$.

## 5.2 Experimental results

We run experiments on different environments cluttered with obstacles. The corresponding videos are available at: http://www.isen.fr/∼sst_lille/fichiers/Icra.wmv.

In these experiments, obstructed areas are created with circular obstacles in the workspace. Some of them are initially out of view from the robots' on-board sensors and may be discovered during the robot movement. The performance criteria is the length of the traveled distance. For the motion planner, the chosen parameters are described in Tab. 1. Furthermore, the parameters for the tracking algorithm are given in Tab. 2. For comparison purpose, the gains $M_1$ and $M_2$ needed to enforce the sliding mode are the same in all the experiments in which they occur. The discontinuous

| | |
|---|---|
| $T_p$ | 2s |
| $T_c$ | 0.5s |
| $n_{\text{knot}}$ | 6 |
| $\varepsilon_v$ | $0.3m/s$ |
| $\varepsilon_w$ | $1rad/s$ |

Table 1: Parameters of the motion planner

| | |
|---|---|
| $\lambda_1$ | 0.5 |
| $\lambda_2$ | 1 |
| $\lambda_3$ | 0.5 |
| $M_1$ | 0.2 |
| $M_2$ | 0.2 |

Table 2: Parameters of the tracking algorithm

controls are approximated by:

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

Figures 10(a) and 11 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 into account the size of the robot, the radius of these obstacles is increased by $0.3m$ (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.

Figures 8, 9 and 10 highlight the performance for the proposed tracking module. The first experiment is made without using feedback controller. By directly applying the open-loop control inputs, the robot does not follow its planned trajectory due to measurement noise and modeling uncertainties. One can see in Fig. 8(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. The third experiment is made using the combination "integral sliding mode controller plus the time-varying state feedback controller". One can note that using only 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 precision motion tracking.

# 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.
- Further, receding horizon planner is a viable method for real time trajectory generation. Depending on computing resources, the use of flatness, B-spline parametrization and constrained feasible sequential quadratic programming can take less than one second to compute an optimal collision-free trajectory.
- Also, 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 modeling errors.

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

# References


[1] J.-P. Laumond, "Robot Motion planning and Control", Springer-Verlag, 1998.

[2] J. C. Latombe, "Robot Motion Planning", Boston: Kluwer Academic Publisher, 1991.

[3] M. Salichs and L. Moreno, "Navigation of mobile robots: open questions", *Robotica*, **18**, pp.227-234, 2000.

[4] Y. Chen and A. Desrochers, "Structure of Minimum-Time Control Law for Robotic Manipulators with Constrained Paths", *IEEE International Conference on Robotics and Automation*, pp. 971-976, 1989.

[5] I. Kolmanovsky and N. H. McClamroch, "Developments in nonholonomic control problems", *IEEE Control Systems Magazine*, **15**, pp. 20-36, 1995.

[6] R. Brockett, "Asymptotic stability and feedback stabilization", in R. Brockett, R. Millman, and H. Sussmann (eds.), Differential geometric control theory (Boston, MA: Birkhauser), pp. 181-195, 1983.

[7] J. Pomet, "Explicit design of time-varying stabilizing control laws for a class of controllable systems without drift", *Systems and Control Letters*, 18(2), pp. 147-158, 1992.

[8] C. Samson, "Control of chained systems: Application to path following and time-varying point-stabilization of mobile robots", *IEEE Trans. on Automatic Control*, **40**, pp. 64-77, 1995.

[9] R. Murray and S. Sastry, "Nonholonomic Motion Planning: Steering Using Sinusoids", *IEEE Trans. on Automatic Control*, **38**(5), pp. 700-716, 1993.

[10] J. P. Hespanha and A. S. Morse, "Stabilization of nonholonomic integrators via logic-based switching", *Automatica*, **35**(3), pp 385-393, 1999.

[11] Z. P. Jiang and H. Nijmeijer, "Tracking control of mobile robots: a case study in backsteeping", *Automatica*, **33**(7), pp. 1393-1399, 1997.

[12] T. Floquet, J. P. Barbot and W. Perruquetti, "Higher-order sliding mode stabilization for a class of nonholonomic perturbed systems", *Automatica*, **39**(6), pp. 1077-1083, 2003.

[13] S. Monaco and D. Normand-Cyrot, "An introduction to motion planning under multirate digital control", *Proceedings of the* $31^{st}$ *Conference on Decision and Control*, pp. 1780-1785, 1992.

[14] C. de Wit and O. Sordalen, "Exponential stabilization of mobile robots with nonholonomic constraints", *IEEE Trans. on Automatic Control*, 37(11), pp. 1791-1797, 1992.

[15] D. Mayne, J. Rawlings, C. Rao and P. Scokaert, "Constrained model predicitive control: Stability and Optimality", *Automatica*, **36**, No 6, pp. 789-814, 2000.

[16] F. A. Cuzzolaa, J. C. Geromel and M. Morari, "An improved approach for constrained robust model predictive control", *Automatica*, **38**, No 7, pp. 1183-1189, 2002.

[17] M. Fliess, J. Lévine, Ph. Martin and P. Rouchon, "Flatness and defect of nonlinear systems: introductory theory and examples", *Int. J. of Control*, **61**(6), pp 1327-1361, 1995.

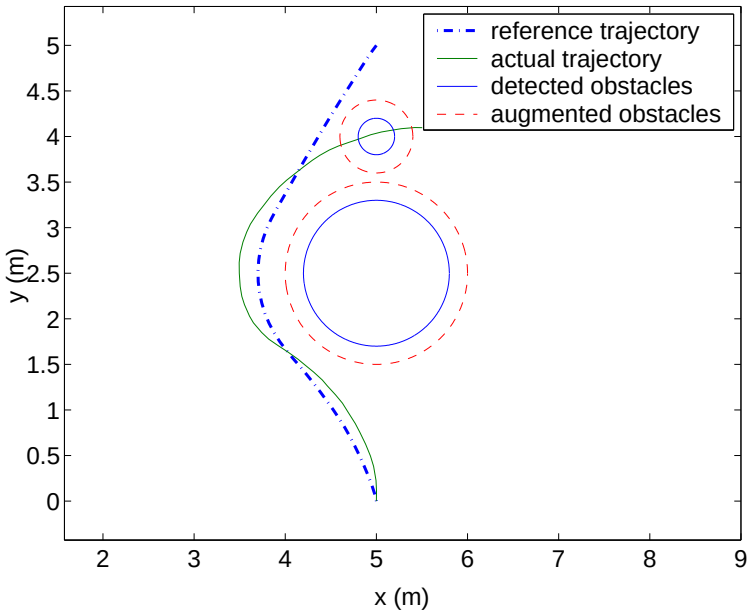
[18] C. Lawrance, J. Zhou and A. Tits, "User's guide for CFSQP Version 2.5", *Institute for Systems Research, University of Maryland, College Park*.

[19] M. B. Milam, "Real time optimal trajectory generation for constrained dynamical systems", *California Institute of Technology*, Dissertation, 2003.

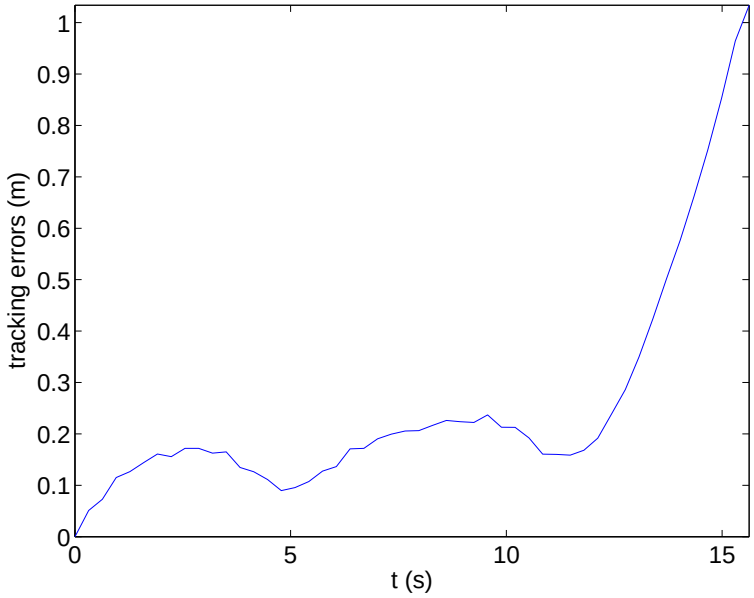
[20] V. Utkin and J. Shi, "Integral sliding mode in systems operating under uncertainty conditions", *Proceedings of the* $35^{th}$ *Conference on Decision and Control*, pp. 4591-4596, 1996.

[21] V. Utkin, "Sliding Modes Control in Electromechanical Systems", Taylor and Francis, 1999.

[22] Z. P. Jiang, E. Lefeber and H. Nijmeijer, "Saturated stabilization and track control of a nonholonomic mobile robot", *Syst. and Cont. Letters*, **42**(5), pp. 327-332, 2001.

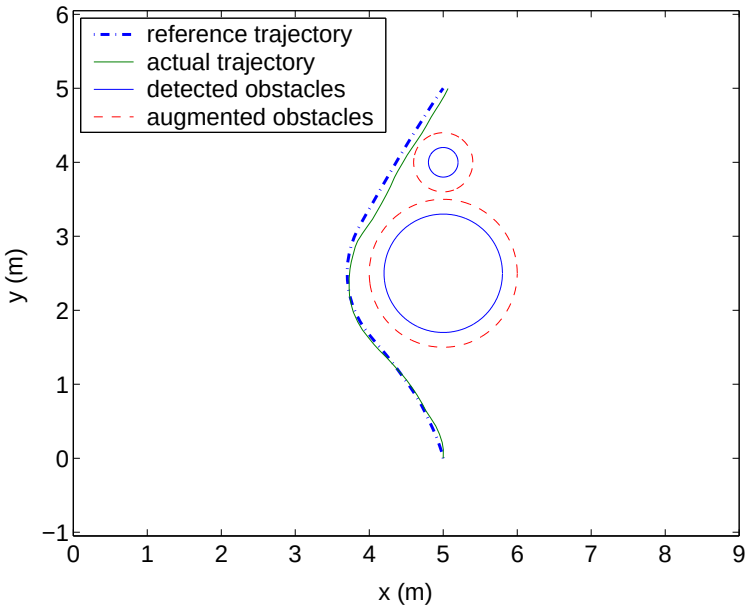



(a)

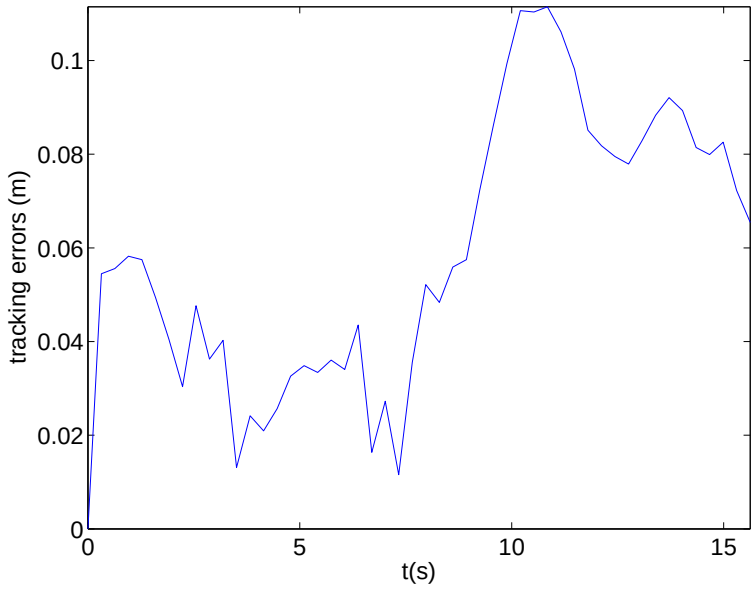



(b)

Figure 8: Experimental results without using the trajectory tracking module

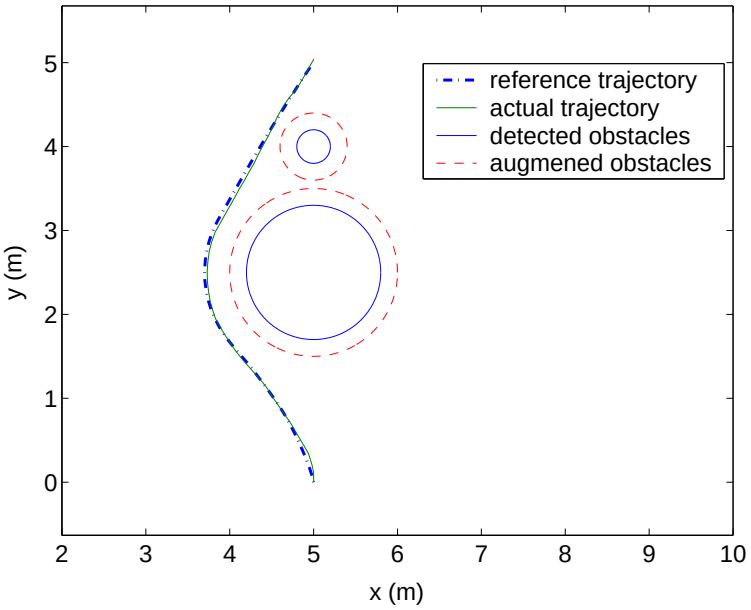



(a)

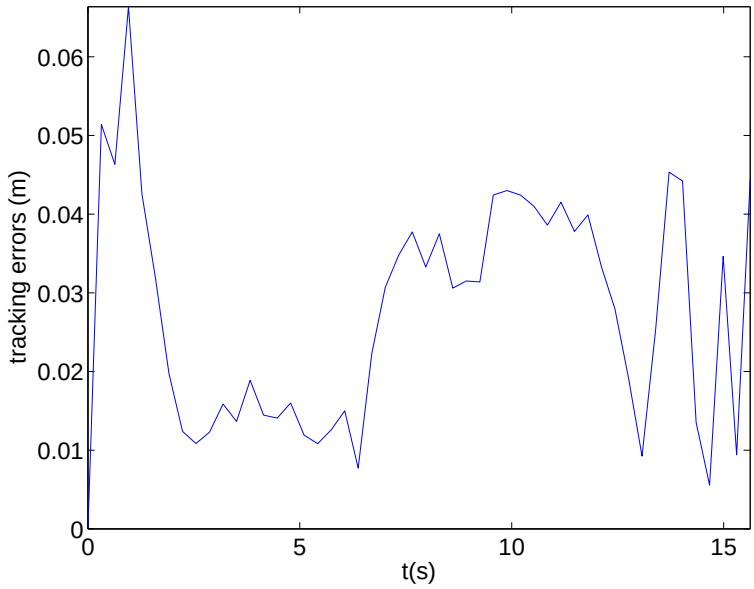



(b)

Figure 9: Experimental results using the time-varying state feedback controller




(a)




(b)

Figure 10: Experimental results using the integral sliding mode and time-varying state feedback controllers

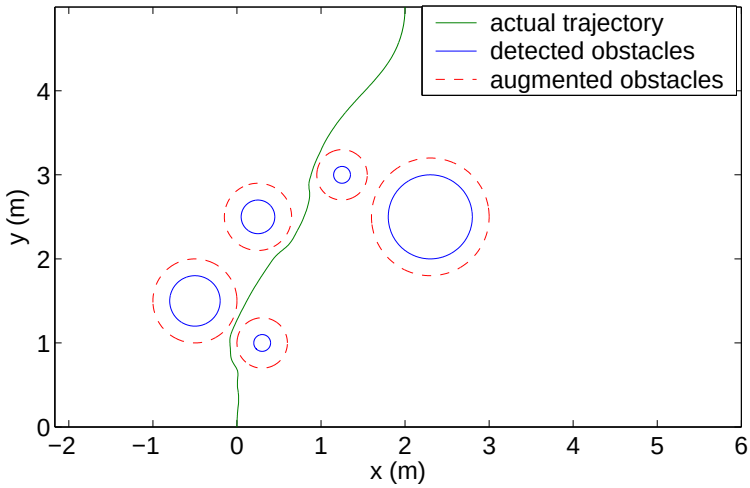



Figure 11: Experimental results using the integral sliding mode and time-varying state feedback controllers in a more complex map