2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

Experimental Motion Planning and Control for an Autonomous Nonholonomic Mobile Robot

Michael Defoort, Jorge Palos, Annemarie Kokosy, Thierry Floquet, Wilfrid Perruquetti and David Boulinguez

Abstract— **This paper presents an architecture for the navigation of an autonomous mobile robot evolving in an uncertain environment with obstacles. The proposed strategy consists in separating the path planning from the control algorithm. The path planning is done by computing the time optimal collision-free trajectory which takes into account the limitations on the linear and angular speeds of the vehicle. The position and shape of obstacles are computed by a vision algorithm using a single camera. A saturated controller based on integral sliding mode is designed to solve the tracking problem in the presence of input saturations and of the unknown disturbances. The effectiveness, perfect performance of obstacle avoidance, real-time and high robustness properties are demonstrated by experimental results.**

I. INTRODUCTION

In real-life environments such as roads or waterways, vehicle use has led to critical problems with respect to cost or safety. Due to the continuous development of sensors and actuators technology, future vehicles might be able to navigate autonomously. In these autonomous vehicles, obstacle avoidance becomes an essential feature.

One difficulty for the control of car-like robots arises from the so-called nonholonomic constraints imposed by the rolling wheels. To safely navigate, the autonomous system must have information about its environment [4], in order to plan its trajectory and finally track it [12]. In this paper, we will focus on the design and implementation of such an architecture. The robot moves among two white lines (a road) with a known distance between them while it avoids static obstacles. The road and the position of obstacles are not known in advance. The robot discovers them as it goes along by using a single camera and infra-red sensors. The overall approach consists of three steps (Fig. 1):

- **Perception using a single camera**. It is the process of transforming measurements of the world into an internal model (position of the obstacles and of the lane, current configuration of the robot, ...). Stereo camera based approaches are expressive enough by using geometrical relationship between cameras but it may be confronted with the matching problem in many cases [1]. Moreover, regarding its costs and the small size of the robot used for the experiments, a single camera is considered as more attractive. The proposed approach is based on the camera calibration and geometrical relationship.
- **Trajectory generation**. It consists in generating a collision-free trajectory from the initial to the final desired positions by solving an optimal control problem subject to dynamic constraints, boundary conditions, trajectory constraints and actuator constraints. Motion planning algorithm that concern obstacle avoidance has been a long standing issue in Robotics. These efforts have concentrated in determining the vehicle paths through free space at the expense of neglecting dynamic constraints or using a simplified version of them [11]. Most of the proposed approaches have theirs inherent limitations. Here, we are interested in the real time implementation of a well known solution based on a direct method [2] and a B-Spline parametrization of the flat outputs [14].
- **Trajectory tracking**. It consists in applying a real time tracking controller to follow the planned trajectory. 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 [3]. To overcome these difficulties, various methods have been investigated: homogeneous and time-varying feedbacks [16], sinusoidal and polynomial controls [15], piecewise controls [8], backstepping approaches [9] or discontinuous controls [7]. However, most of these methods do not provide good robustness properties or do not take into account the important saturation constraints on control inputs.
  This objective will be achieved by using integral sliding mode control [5], [17]. A drawback of the *classical* sliding mode control is that the trajectory of the designed solution is not robust on a time interval preceding the sliding motion. Here, the design and integration of an integral sliding mode controller which takes into account the saturation constraints on velocities are presented.

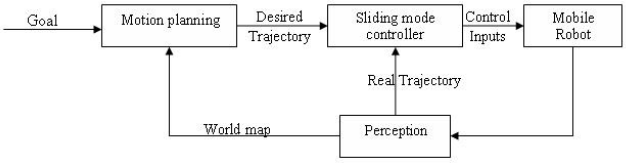


Fig. 1. Architecture of the autonomous navigation

This work was partially supported by programs Interreg **ACOS**, TAT T31 and ARCIR **Robocoop**.

M. Defoort, T. Floquet and W. Perruquetti are with LAGIS UMR CNRS 8146, Ecole Centrale de Lille, BP 48, Cite Scientifique, 59 651 Villeneuve-d'Ascq, France

J. Palos, A. Kokosy and D. Boulinguez are with ISEN, 41 bvd Vauban, 59 046 Lille Cedex, France

The main contribution of this paper is the integration of a complete autonomous navigation architecture in a mobile robot. The outline of this paper is as follows. The vision algorithm is detailed in section II. Section III gives the planning scheme. Then, the integral sliding mode controller is presented in section IV. Finally, in section V, we present the integration of the different modules and the experimental results on a car-like robot.

II. VISION ALGORITHM

In our application, the sensors used to observe the surroundings are infra-red sensors and a single video camera. Note that the infra-red sensors provide information about the presence of obstacles and the video camera is used in order to extract the useful information for the motion planning phase (positions of lane and obstacles).

A. Pre-processing : lane and obstacle detection

The first step is the extraction of various characteristic points of the image which gives enough information for the reconstruction of the scene. A suitable thresholding enables to extract the white lanes from the image, and another one allows to find the objects whose colors differ from the one of the image background. Then, the morphological filtering (morphological closing and morphological opening) enables to eliminate some residual objects compared to the thresholdings. Finally, the image labelling of the resultant image enables to treat each object in a different way, and thus to separate them from the image.

B. Reconstruction of the real scene

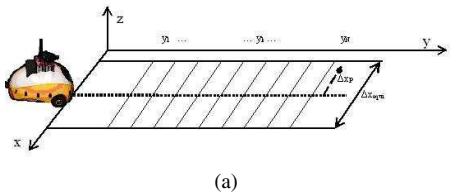
A $3D$ description of the world using a single image is impossible without assumptions and calibration. In order to associate each pixel of the image with the corresponding point in $3D$ coordinates, the following assumptions are done:

- The road is flat
- The position of the camera is fixed on the robot. In the robot coordinate system, the camera has a constant coordinate (x_c, y_c, z_c).

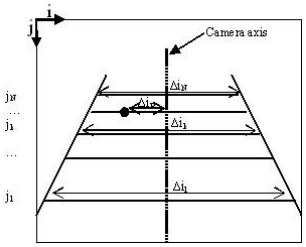
Since the road is assumed to be flat, the robot moves on a surface where $z = 0$. Therefore, one will rather speak about $2D+$ coordinates. Two spaces will be used:

- the $2D+$ world space $W = \{x, y, 0\}$, with $x, y \in \mathbb{R}$,
- the $2D$ image space $I = \{i, j\}$ with $i, j \in \mathbb{N}$.

The camera calibration enables to have a view of the scene in front of the robot by removing the perspective effect induced by the acquisition conditions. Since the camera is fixed on the robot, one can establish a relationship between the pixels of the image acquired by this camera and the $2D+$ coordinates in the camera coordinate system. Then, by using elementary geometrical transformations, it will be easy to compute the $2D+$ coordinates in the real reference coordinate system.



(a)



(b)

Fig. 2. Camera calibration: lane detection

1) Lane detection : relationship between coordinates $(\Delta x_P, \Delta y_P)$ of a point of the road in the camera coordinate system and image I: First, a correspondence grid is determined between:

- each ordinate Δy_k, $k \in \{1, \ldots N\}$ in the camera coordinate system,
- each ordinate j_k in I.

It is generated for the camera field of view by spacing measurements in an equidistant way (see Fig. 2). Thanks to this grid, the ordinate Δy_P can be determined by measuring the ordinate j_P of the associated point on image I. All lines $(i, j_k)|_{k\in\{1,\ldots N\}}$ of length δi_k are associated with Δx in W. Therefore, one can easily deduce the abscise Δx_P of a real point by applying: $\Delta x_P = \frac{\Delta x \delta i_P}{\delta i_k}$.

2) Obstacle detection : relationship between the position $(\Delta x_o, \Delta y_o)$ of the obstacle in the camera coordinate system and image I: It is assumed that the shape of the obstacles is known. As for the lane detection, a correspondence grid is first determined between:

- the ordinate Δy_o of the position of the obstacle in the camera coordinate system,
- the width of the objects in I

It is well known that the size of an object in I exponentially decreases with respect to Δy_o. Then, the distance between the center of the object and the camera axis in I provides the position Δx_o of the obstacle (Fig. 3).

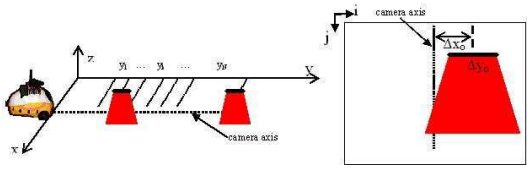


Fig. 3. Camera calibration: obstacle detection

3) Geometrical transformation: Knowing the position (x,y) and the orientation θ of the robot, the coordinates of one

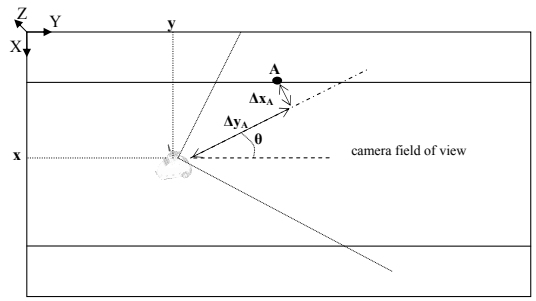


Fig. 4. Geometrical transformation

point (x_A, y_A) in the real coordinate system can be expressed with its coordinate $(\Delta x_A, \Delta y_A)$ on the camera coordinate system by (see Fig. 4):

$$\begin{bmatrix} x_A \\ y_A \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \Delta x_A \\ \Delta y_A \end{bmatrix} + \begin{bmatrix} x + x_c \\ y + y_c \end{bmatrix}$$

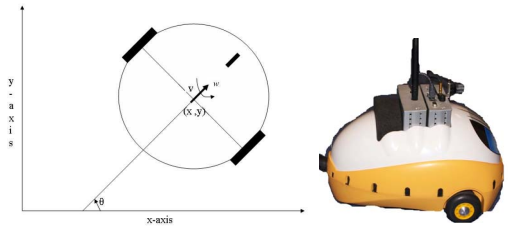
III. Motion planning

Once a consistent world model is created in real time, a trajectory which fulfills collision avoidance and computation time constraint is generated.

A. Notations

The robot is represented by a disc located at center of mass (x, y) and of radius r (Fig. 5). $q = [x, y, \theta]^T$ and $U = [v, w]^T$ respectively denote the state variables and the control inputs (linear and angular velocities). The kinematic equations of the system under the nonholonomic constraint of pure rolling and no slipping can be written as follows:

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



(a) (b)

Fig. 5. The car like robot

B. Perception and motion planning cycle

The proposed motion planning explicitly takes into account the real time constraint imposed by the environment. The mobile robot, located in an unknown environment, has a limited time to compute its desired trajectory. The time δt to make its decision depends on its real time environment. The time t_i ($i \in \mathbb{N}$) of the updated environment depends on the information of the infra-red sensors. Indeed, when the mobile robot detects an obstacle which may cause a collision, using the infra-red sensors, a cycle of perception and motion planning starts. A cycle (see Fig. 6), starting at time t_i, is described as follows:

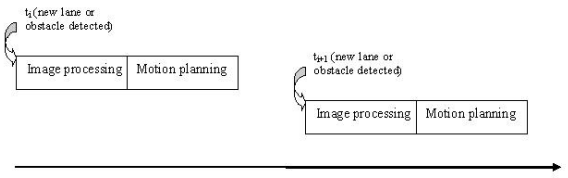


Fig. 6. Perception and motion planning cycles

1) Update the model of the environment by using a camera, i.e. position (x_m^o, y_m^o) of the detected obstacles ($m = 1, \ldots N_o$, N_o is the number of detected obstacles) and position of the detected lane. The obstacles are assumed to be circular with radius r_m^o.
2) Determine the feasible quasi-minimum time trajectory based on sequential quadratic programming [13].
3) At time $t_i + \delta_t$, the current iteration is over. The best feasible trajectory is selected.

C. Motion planning algorithm

The goal is to generate a feasible quasi-minimum time trajectory that satisfies the environmental constraints: limits of the lane, obstacle avoidance and also the physical constraints due to the limitations on the velocities. The optimal control problem is to find the control inputs which minimize:

$$J = \int_{\tau_i}^{t_f} dt, \tag{2}$$

where the initial time is $\tau_0 = 0$, $\tau_i = t_i + \delta t$ and t_f is the unknown final time. The trajectory must join the known states $q(\tau_i)$, $q(t_f)$ and satisfies the constraints, $\forall t \in [\tau_i, t_f]$:

$C1$ the control bounds:

$$|v| \le v_{max} - \varepsilon_v, \quad |w| \le w_{max} - \varepsilon_w,$$

where ε_v and ε_w are positive control parameters. The inclusion of these constants in the constraints of the motion planning generator guarantees that there is sufficient control authority to track the trajectory.

$C2$ the limits of the lane $\mathscr{L}$:

$$(x, y) \in \mathscr{L}$$

$C3$ the collision avoidance with the N_o detected obstacles:

$$\forall m \in \{1, \ldots, N_o\}, \ \sqrt{(x - x_m^o)^2 + (y - y_m^o)^2} \ge r + r_m^o$$

Using the flatness property [6] of system (1), all system variables can be differentially parameterized by x, y and a finite number of their time derivatives. Indeed, θ, v and w can be expressed by x, y and their first and second time derivatives, i.e.

$$\theta = atan\frac{\dot{y}}{\dot{x}}, \quad v = \sqrt{\dot{x}^2 + \dot{y}^2}, \quad w = \frac{\ddot{y}\dot{x} - \ddot{x}\dot{y}}{\dot{x}^2 + \dot{y}^2}. \tag{3}$$

Once the constraints C1-C3 are mapped into the flat output space, the time optimal trajectory is planned in this space. First, for each flat outputs i.e x and y, an initial guess trajectory ($x_{ini}(t)$ and $y_{ini}(t)$) which respects the boundary constraints is computed.

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. 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. The time interval of $[\tau_i, t_f]$ is divided into $P+1$ equal segments with $P+4$ knots to be control knots:

$$q_0 = q_1 = \tau_i < q_2 < \ldots < q_{P+2} = t_f = q_{P+3} \tag{4}$$

The trajectories of the flat outputs are defined as:

$$\begin{array}{rcl} x(t) & = & x_{ini}(t) + \sum_{j=1}^{P} a_j B_{j,3}(t), \\ y(t) & = & y_{ini}(t) + \sum_{j=1}^{P} b_j B_{j,3}(t) \end{array} \tag{5}$$

where $a_j, b_j \in \mathbb{R}$, $\mathscr{P} = \{a_1, \ldots, a_P, b_1, \ldots, b_P\}$ is the parameter vector and $B_{j,3}$ is the B-spline basis function computed recursively as follows:

$$\begin{array}{rcl} B_{j,0}(t) & = & \left\{ \begin{array}{l} 1 \text{ if } q_j \le t < q_{j+1} \\ 0 \text{ otherwise} \end{array} \right. \\ \forall d & \in & \{1,2,3\}, \\ B_{j,d}(t) & = & \dfrac{t-q_j}{q_{j+d}-q_j} B_{j,d-1}(t) + \dfrac{q_{j+d+1}-t}{q_{j+d+1}-q_{j+1}} B_{j+1,d-1}(t) \end{array} \tag{6}$$

Finally, let us consider the uniform time partition:

$$\tau_i = \eta_0 < \eta_1 < \ldots \eta_{N_{sample}-2} < \eta_{N_{sample}-1} = t_f \tag{7}$$

where N_{sample} is the number of sampled times. The optimal control problem can be rewritten in a discrete way as follows:

$$\min_{\mathscr{P}} (t_f - \tau_i) \tag{8}$$

subject to: $\forall m \in \{1, \ldots, N_o\}$ and $\forall i \in \{0, \ldots, N_{sample}-1\}$

$$\begin{array}{rcl} \sqrt{\dot{x}(\eta_i)^2 + \dot{y}(\eta_i)^2} & \le & v_{max} - \varepsilon_v, \\ \left| \frac{\ddot{y}(\eta_i)\dot{x}(\eta_i) - \ddot{x}(\eta_i)\dot{y}(\eta_i)}{\dot{x}(\eta_i)^2 + \dot{y}(\eta_i)^2} \right| & \le & w_{max} - \varepsilon_w, \\ (x(\eta_i), y(\eta_i)) & \in & \mathscr{L} \\ \sqrt{(x(\eta_i) - x_m^o)^2 + (y(\eta_i) - y_m^o)^2} & \ge & r + r_m^o \end{array} \tag{9}$$

The optimal coefficients $\mathscr{P}$ are computed using the CFSQP optimization algorithm. To finish, the open loop control inputs are deduced using equation (3).

IV. Sliding mode controller for trajectory tracking with saturation constraints

A. Formulation of the tracking problem

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

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{bmatrix} = \begin{bmatrix} \cos\theta_r & 0 \\ \sin\theta_r & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_r \\ w_r \end{bmatrix} \tag{10}$$

where the desired velocities v_r and w_r satisfy:

$$|v_r| \le v_{max} - \varepsilon_v, \quad |w_r| \le w_{max} - \varepsilon_w$$

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

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v + \delta_v \\ w + \delta_w \end{bmatrix} \tag{11}$$

where δv and δw represent the uncertainties. Due to saturation constraints, it is assumed that:

$$|\delta v| < \varepsilon_v, \quad |\delta w| < \varepsilon_w$$

Controls v and w must be designed such that system (11) follows reference (10) 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 following constraints:

$$|\, v \,| \le v_{max}, \quad |\, w \,| \le w_{max}. \tag{12}$$

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 model is represented by:

$$\begin{bmatrix} \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \end{bmatrix} = \begin{bmatrix} v_r \cos e_3 \\ v_r \sin e_3 \\ w_r \end{bmatrix} + \begin{bmatrix} -1 & e_2 \\ 0 & -e_1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} v + \delta_v \\ w + \delta_w \end{bmatrix}.$$

So, the tracking dynamics can be described as

$$\dot{e} = f_1(e) + f_2(e)(U + \delta) \tag{13}$$

with $e = [e_1, e_2, e_3]^T$, $U = [v, w]^T$ and $\delta = [\delta_v, \delta_w]^T$.

B. Integral sliding mode controller

For system (13), the control law is defined as follows:

$$U = U_0 + U_1. \tag{14}$$

U_0 is the ideal control and U_1 represents the ISM part which is designed to be discontinuous in order to reject the perturbation.

The first part of the control design is to find a saturated control law U_0 such that the nominal system $\dot{e} = f_1(e) + f_2(e)U_0$ is globally asymptotically stable. The chosen control law U_0 has been developed by Jiang in [10]. The motivation for such a choice is that this design takes into account the actuator bounds. It is described by:

$$U_0 = \begin{bmatrix} v_0 = v_r \cos e_3 + \lambda_3 \tanh e_1 \\ w_0 = w_r + \frac{\lambda_1 v_r e_2}{1 + e_1^2 + e_2^2} \frac{\sin e_3}{e_3} + \lambda_2 \tanh e_3 \end{bmatrix} \tag{15}$$

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:

$$|v_0| \leq v_{max} + \lambda_3, \quad |w_0| \leq w_{max} + \frac{\lambda_1 v_{max}}{2} + \lambda_2$$

Remark 1: One can note that using only controller (15), the robustness performance is not good enough. Therefore, in order to improve the robustness properties, a discontinuous term based on integral sliding mode control is added to this existing controller.

Let define the sliding variable s as:

$$s = [s_1, s_2]^T = s_0(e) + z. \tag{16}$$

where

$$\begin{array}{rcl} s_0(e) & = & [-e_1, -e_3]^T, \\ \dot{z} & = & -\frac{\partial s_0}{\partial e}(f_1(e) + f_2(e)U_0), \\ z(0) & = & -s_0(e(0)) \end{array} \tag{17}$$

Here, z induces the integral term and provides one more degree of freedom in the sliding variable design. Initial condition $z(0)$ is determined such that the sliding variable always satisfies $s(0) = 0$. Hence, the controlled system slides on the sliding surface $\{s = 0\}$ from the initial time instant without any reaching phase.

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 manifold.

$$\begin{array}{rcl} \dot{V} & = & s^T \left(\frac{\partial s_0}{\partial e}\dot{e} - \frac{\partial s_0}{\partial e}(f_1(e) + f_2(e)U_0) \right) \\ & = & \left(\left[\frac{\partial s_0}{\partial e} f_2(e) \right]^T s \right)^T (U_1 + \delta) < 0 \end{array}$$

The above condition holds if:

$$U_1 = \begin{bmatrix} -M_1 sign(s_1) \\ -M_2 sign(-e_2 s_1 + s_2) \end{bmatrix} \tag{18}$$

with $M_1 > \delta_v + \mu$ and $M_2 > \delta_w + \mu$ $(\mu > 0)$.

Remark 2: The trajectory evolves on the manifold $s = 0$ from $t = 0$ and remains there in spite of the perturbations. The time derivative of the sliding variable is $\dot{s} = \frac{\partial s_0}{\partial e}(\dot{e} - f_1(e) - f_2(e)U_0) = 0$. Therefore, the motion equation in sliding mode is $\dot{e} = f_1(e) + f_2(e)U_0$ which is globally asymptotically stable.

Remark 3: 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.

V. Experimental results

A. Experimental setup

The proposed motion planning and control algorithms were implemented on the mobile robot Pekee manufactured at Wany Robotics company. An Intel 486 micro-processor running at $75MHz$ operating under linux real time hosts the integral sliding mode controller written in C. Pekee is equipped with 15 infra-red telemeters sensors, two encoders, a WiFi wireless cartridge and a miniature color vision camera C-Cam8. The vision camera is fixed in the robot coordinate system $(x_c, y_c, z_c) = (0, 0, 0.25)$. The maximum linear and angular speeds are respectively equal to $v_{max} = 0.35$m/s and $w_{max} = 0.8$rad/s. The computing time δt including the image processing and the motion planning algorithm is about five minutes on the embedded $75MHz$ PC. In order to decrease the computing time, we used the socket protocol communication and Wifi. The image data are sent to a Pentium IV $1.7GHz$ PC for the image processing and for the generation of the time optimal trajectory. This protocol enabled to reduce the computing time δt to $3s$.

B. Experimental results

The following parameters are chosen for the motion planning algorithm (4)-(9): $P = 10$, $\varepsilon_v = 0.5$m/s, $\varepsilon_w = 0.2$rad/s, $N_{sample} = 100$. The integral sliding mode controller parameters are set to $\lambda_1 = 0.2$, $\lambda_2 = 0.1$, $\lambda_3 = 0.4$, $M_1 = 0.1$ and $M_2 = 0.1$.

The map of obstacles of radius 0.3m, given in Tab. I, is not initially known and will be discovered during the robot movement. In Tab. I, the positions of detected obstacles are given. One can see that the image procession module gives good results. The initial and final configurations of the robot are given in Tab. II. The two white lines of the lane are straight and fulfill equations $x = 0$ and $x = 10$.

TABLE I
POSITIONS OF OBSTACLES

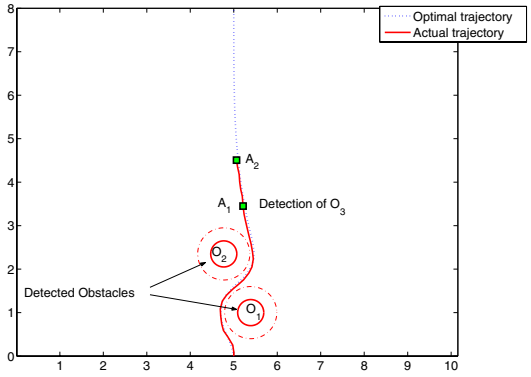
Real positions	$x_1^o = 5.45$m	$x_2^o = 4.80$m	$x_3^o = 5.30$m
	$y_1^o = 0.90$m	$y_2^o = 2.40$m	$y_3^o = 5.45$m
Detected positions	$x_1^o = 5.39$m	$x_2^o = 4.77$m	$x_3^o = 5.24$m
	$y_1^o = 1.00$m	$y_2^o = 2.35$m	$y_3^o = 5.49$m

TABLE II
TERMINAL CONFIGURATIONS OF THE ROBOT

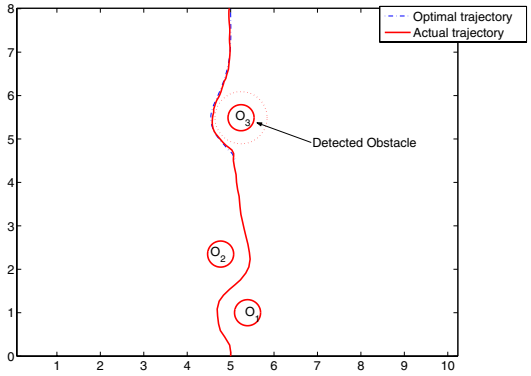
$x(0) = 5$m	$y(0) = 0$m	$\theta(0) = \pi/2$rad
$x(t_f) = 5$m	$y(t_f) = 8$m	$\theta(t_f) = \pi/2$rad

Fig. 7 depicts the planned and executed trajectories in the unknown map. First, the robot visualizes the scene and applies the image processing. Obstacles O_1 and O_2 are detected. In order to take into account the image processing errors and for safety consideration, the radius of these obstacles is increased by 0.3m (dotted lines around obstacles). According to the detected obstacles, a collision free trajectory using the motion planning generator is planned (see Fig. 7.a). Then, the integral sliding mode controller enables to track the desired trajectory in spite of uncertainties and errors. During the execution, the robot may detect possible obstacles using its

infra-red sensors. These sensors inform if there are obstacles in its planned trajectory. If an obstacle is detected (see Fig. 7.a), a new perception and motion planning cycle begins while the robot moves from A_1 to A_2. When the robot reaches at A_2, the robot tracks the new planned trajectory (see Fig. 7.b).



(a)



(b)

Fig. 7. Actual trajectory of the mobile robot

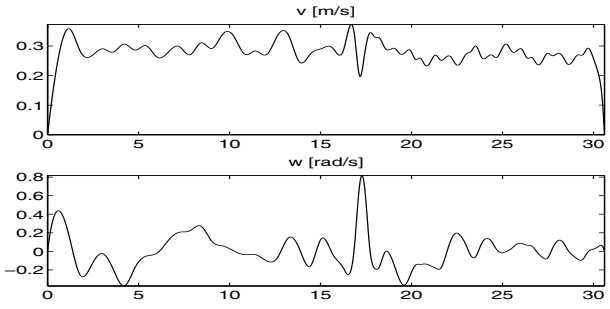


Fig. 8. Apply control inputs v and w

VI. Conclusions and Future Work

An architecture to the navigation of an autonomous mobile robot evolving in an uncertain environment with obstacles is proposed. The perception algorithm provides an accurate localization of the lane and the detected obstacles. Then, the planning algorithm generates a time optimal feasible trajectory. Finally, an integral sliding mode controller is presented in order to provide high robustness properties. Our architecture was implemented on a robot and provides real time, high robustness properties and good performance for obstacle avoidance.

References

[1] M. Bertozzi and A. Broggi, "GOLD: A parallel real-time stereo vision system for generic obstacle and lane detection", *IEEE Trans. On Image Processing*, **7**(1), pp. 62-81, 1998.

[2] J. T. Betts, "Survey of Numerical Methods for trajectory optimization", *J. of Guidance, Control and Dynamics*, **21**(2), pp. 193-207, 1998.

[3] 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.

[4] G. N. DeSouza and A. C. Kak, "Vision for Mobile Robot Navigation: A Survey", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **24**, pp. 237–267, 2002.

[5] M. Defoort, T. Floquet, A. Kokosy and W. Perruquetti, "Integral sliding mode control for trajectory tracking of a unicycle type mobile robot", *Integrated Computer Aided Engineering*, **13**, pp. 277-288, 2006.

[6] 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.

[7] 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.

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

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

[10] 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.

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

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

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

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

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

[16] 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.

[17] 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.