Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online
CONFERENCE PAPER ARCHIVE

IFAC-PapersOnLine 49-15 (2016) 013–018

Kinematic trajectory tracking controller for an all-terrain Ackermann steering vehicle

Luca Bascetta * **Davide A. Cucci** ** **Matteo Matteucci** *

* *Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria, Piazza Leonardo da Vinci 32, 20133 Milano, Italy (e-mail: {luca.bascetta, matteo.matteucci}@polimi.it)*
** *École Polytechnique Fédérale de Lausanne, Geodetic Engineering Laboratory, Station 18, CH-1015 Lausanne, Switzerland (e-mail: davide.cucci@epfl.ch)*

Abstract: In the last few years the number of applications of autonomous mobile robots to outdoor and off-road tasks, like border surveillance and monitoring, search and rescue, agriculture, driveless mobility has been rapidly increasing. Among the huge number of functionalities that are required to make a vehicle an autonomous robot, localisation, path planning and trajectory tracking are the most important. This paper proposes a novel approach to solve the trajectory tracking problem of an Ackermann steering vehicle. A multi-body dynamic model of the vehicle is proposed as a mean to tune and validate the tracking controller. The proposal is supported by an experimental validation, that shows the effectiveness of the trajectory tracking controller and its performance in comparison with the accuracy of the localisation algorithm.

Keywords: Autonomous offroad vehicles, Kinetic control system, Trajectory tracking, Non-holonomous vehicles, Nonlinear control.

1. INTRODUCTION

The popularity of the research on wheeled mobile robots has been recently increasing, due to their possible use in different outdoor environments. Planetary explorations, search and rescue missions in hazardous areas, surveillance, humanitarian de-mining, driveless mobility, as well as agriculture works such as pruning vine and fruit trees, represent possible applications for autonomous vehicles in natural and urban environments. Differently from the case of indoor mobile robotics, where only flat terrains are considered, outdoor robotics deals with all possible natural terrains. The unstructured environment, terrain roughness and poorly traversable terrains, that characterise a natural environment, or the dynamic obstacles and safety issues, that characterise an urban scenario, make the development of an autonomous vehicle a challenging problem.

Among the huge number of functionalities that are required to develop an autonomous vehicle, three are particularly important, i.e. localisation, path planning, including obstacle avoidance, and trajectory tracking. Though they are all crucial to let the vehicle complete a task, it must be noticed that the behaviour of the vehicle is directly related to the trajectory tracking controller and to the interaction between this system and the localisation module. Further, in natural outdoor environments, characterised by poorly traversable terrains, slopes and a variety of obstacles, the accuracy of the trajectory tracking algorithm is extremely important in order to ensure the safety of the vehicle, as the more secure path should be the one generated by the planner, any deviation from this path can lead the vehicle to a potentially dangerous situation. The same reasoning obviously applies to urban environments, where the accuracy of the trajectory tracking is even more important, as any deviation from the planned path leads to a potentially dangerous situation for the humans close to the vehicle.

In the last decade many researchers have focused their attention on the path and trajectory tracking control problems, either for indoor and outdoor environments and for different vehicle kinematics. In particular, in Aicardi et al. (1995) the authors show that the limitation imposed by Brockett theorem can be overcame with a special choice of the system state variables, guaranteeing global stability for the trajectory tracking of a unicycle-like vehicle by a smooth feedback control law. The same authors, in Balluchi et al. (1996) proposed a path tracking controller based on the sliding mode technique for Dubin's car. Differently from the previous approaches, in Indiveri (1999) a time-invariant controller to solve the parking problem for a bicycle-like kinematic is presented. Recently, researches have considered again the path tracking problem (Kim and Minor (2005, 2008); Sgorbissa and Zaccaria (2009); Morro et al. (2011)) focusing their attention, however, on unicycle-like vehicles.

This work proposes a novel trajectory tracking control law that is inspired by Aicardi et al. (1995) and Indiveri (1999). Following Indiveri (1999) a control law to solve the parking problem, i.e., to make the vehicle asymptotically converge to a desired pose, is here derived. Then, this law is extended in order to support trajectory tracking. This extension follows the approach already presented in Aicardi et al. (1995), making it applicable to an Ackermann steering kinematics. Summarising, the contribution of this paper with respect to Aicardi et al. (1995) and Indiveri (1999), where a tracking problem on a unicycle-like vehicle and a parking problem on a bicycle-like vehicle have been considered, is a nonlinear controller that merges the theory developed in the aforementioned papers

in order to devise a methodology for trajectory tracking of Ackermann steering kinematics.
Furthermore, a procedure to support and simplify the selection of the controller parameters, relating their values to the desired trajectory tracking error, is also introduced.
The work is completed by an experimental validation of the proposed control law on a commercial all-terrain vehicle.

2. A KINEMATIC TRAJECTORY TRACKING CONTROLLER

As far as path/trajectory tracking for an outdoor autonomous vehicle is concerned, the control law has to fulfil some important features.

First of all, it must ensure the stability of the closed loop system, whatever the pose of the vehicle with respect to the desired path is. Then, the control system should automatically manage either the trajectory tracking phase, i.e. when the vehicle is rather close to the desired trajectory, and the approach to the path from a pose far from it, always ensuring smooth variations in the control signals. This characteristic is of particular importance, as in case of an obstacle suddenly appearing in front of the vehicle an emergency action could be triggered driving it far from the planned path.
Furthermore, the control law should be simple and easily implementable, and it should not require the knowledge of the vehicle dynamic parameters, as their identification can be complex and time consuming, or they even need to be on-line estimated in order to cope with time variability. Nevertheless, the controller has to accommodate for the characteristics of a real vehicle.

2.1 The trajectory tracking control law

An Ackermann steering vehicle can be suitably represented, for the sake of designing the trajectory tracking controller, by a kinematic bicycle model (Figure 1).
Considering the centre of the rear wheel of the bicycle, its motion is described by the following equations

$$\begin{aligned} \dot{x} &= v\cos\phi \\ \dot{y} &= v\sin\phi \\ \dot{\phi} &= v\frac{\tan\psi}{\ell} \end{aligned} \tag{1}$$

where (x, y, ϕ) define the vehicle pose, i.e. the position and orientation of the rear wheel-frame in the Cartesian plane, v the vehicle linear velocity, ψ the steering angle and ℓ the length of the vehicle.

Introducing now a desired vehicle pose (x_d, y_d, ϕ_d), a set of polar-like variables can be derived Indiveri (1999)

$$\begin{aligned} e &= \sqrt{(x_d-x)^2+(y_d-y)^2} \\ \theta &= \operatorname{atan2}(y_d-y, x_d-x)-\phi_d \\ \alpha &= \theta-\phi+\phi_d \end{aligned}$$

and equation (1) can be expressed in polar coordinates as it follows

$$\begin{aligned} \dot{e} &= -v\cos\alpha \\ \dot{\alpha} &= v\frac{\sin\alpha}{e} - v\frac{\tan\psi}{\ell} \\ \dot{\theta} &= v\frac{\sin\alpha}{e} \end{aligned} \tag{2}$$

Following Indiveri (1999) we can now synthesise a control law for the so-called parking problem, i.e., driving the vehicle

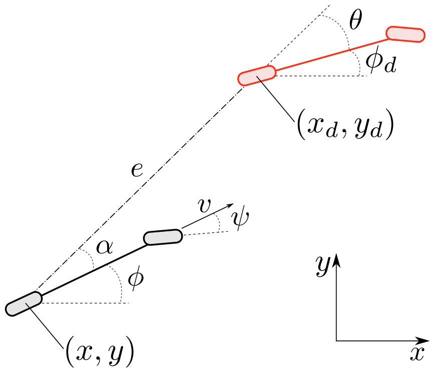

Fig. 1. Bicycle kinematic model (in black the actual vehicle position, in red the desired one).

to asymptotically converge to a desired pose. First, assuming that the vehicle can move only in the forward direction[1], a proportional controller can be designed for the linear velocity v as it follows

$$v = \gamma e, \quad \gamma > 0 \tag{3}$$

Introducing this equation into the kinematic model (2), it follows

$$\begin{aligned} \dot{e} &= -\gamma e\cos\alpha \\ \dot{\alpha} &= \gamma e\left(c - \frac{\sin\alpha}{e}\right) \\ \dot{\theta} &= \gamma\sin\alpha \end{aligned}$$

where

$$c = \frac{\tan\psi}{\ell}$$

is the curvature.
The last equations, together with the quadratic function

$$V = \frac{1}{2}\left(\alpha^2 + h\theta^2\right), \quad h > 0$$

whose time derivative, along the trajectories of the system, is given by

$$\dot{V} = \alpha\dot{\alpha} + h\theta\dot{\theta} = \gamma(\alpha\sin\alpha + h\theta\sin\alpha - \alpha e c)$$

suggest to choose the curvature as it follows

$$c = \frac{\sin\alpha}{e} + h\frac{\theta}{e}\frac{\sin\alpha}{\alpha} + \beta\frac{\alpha}{e}, \quad \beta > 0 \tag{4}$$

Substituting now equation (4) into the derivative of V it follows

$$\dot{V} = -\gamma\beta\alpha^2 \leq 0$$

Then, it can be shown that e, α and θ asymptotically exponentially converge to zero (see Indiveri (1999) for further details).

The behaviour of the closed loop system depends on the selection of the parameters γ, β and h that appears in (3) and (4). Though γ can be easily chosen, making reference to the vehicle dynamics and actuator saturation, the selection of β and h is more involved. It can be shown (see Indiveri (1999) for further details), however, that satisfying the following constraints

$$h > 1 \quad 2 < \beta < h + 1$$

the vehicle approaches the target along a straight line.

The control laws (3) and (4) allow to asymptotically converge to a desired pose, but they are not suitable to drive the vehicle along a desired trajectory. For this reason, the approach

[1] An ATV can move in forward and backward direction but, in order to shift it into reverse, it needs to stop and activate the brakes. For this reason, at trajectory tracking level it is reasonable to assume that the vehicle can move only in the forward direction.

in Aicardi et al. (1995) is now extended in order to support trajectory tracking. The trajectory tracking controller is derived modifying the methodology presented in Indiveri (1999) in order to be compliant with an Ackermann steering kinematics.

Assume that the desired trajectory can be parametrised using the natural coordinate s. Chosen an initial position p_o on the path, an orientation, and a velocity $\dot{s} \geq 0$, $p(s)$ denotes the target frame tangent to the trajectory.
The kinematic model (2) can be modified, in order to take into account that the vehicle target pose is now moving along the desired trajectory, as it follows

$$\begin{aligned}\dot{e} &= -v\cos\alpha + \dot{s}\cos\theta \\ \dot{\alpha} &= v\frac{\sin\alpha}{e} - \dot{s}\frac{\sin\theta}{e} - v\frac{\tan\psi}{\ell} \\ \dot{\theta} &= v\frac{\sin\alpha}{e} - \dot{s}\frac{\sin\theta}{e} - \frac{\dot{s}}{R(s)}\end{aligned} \tag{5}$$

where $R(s)$ is the signed curvature radius at the target position. Substituting now the previous control laws into the kinematic model (5), the following closed loop systems is obtained

$$\begin{aligned}\dot{e} &= -\gamma e\cos\alpha + \dot{s}\cos\theta \\ \dot{\alpha} &= -\dot{s}\frac{\sin\theta}{e} + \gamma h\frac{\sin\alpha}{\alpha}\theta + \gamma\beta\alpha \\ \dot{\theta} &= \gamma\sin\alpha - \dot{s}\frac{\sin\theta}{e} - \frac{\dot{s}}{R(s)}\end{aligned}$$

where $\dot{s}$ is now the only input that can be used in order to enforce the trajectory tracking.
In accordance with Aicardi et al. (1995), the target velocity can be specified on-line as

$$\dot{s} = \begin{cases} 0 & V = \lambda e^2 + \left(\alpha^2 + h\theta^2\right) > \varepsilon \\ f(e,\alpha,\theta) & V = \lambda e^2 + \left(\alpha^2 + h\theta^2\right) \leq \varepsilon \end{cases} \quad 0 < \varepsilon < \frac{\pi^2}{4} \tag{6}$$

where $f(e,\alpha,\theta)$ is any continuous radial function centred on the ellipsoidal domain $\lambda e^2 + \left(\alpha^2 + h\theta^2\right) = \varepsilon$.
Equation (6) allows for the following interpretation. When the vehicle position is outside the ellipsoidal domain V the target position is kept constant and a parking problem is activated in order to move the vehicle towards the target. As previously noticed, in this case e, α and θ will asymptotically exponentially converge to zero.
On the other side, when the vehicle is inside the ellipsoid, $\dot{s} \neq 0$ and, consequently, $e \neq 0$ causing its motion towards the target position. Though during this phase the convergence of the error to zero cannot be guaranteed, it is straightforward that if the error increases the vehicle exits the ellipsoidal domain and a new parking problem is triggered (for further details see Aicardi et al. (1995)).

In order to guarantee that the vehicle follows the path at the desired speed v_d, and the motion is sufficiently smooth – avoiding a continuous switching between a parking and a following problem, even in the presence of errors in the vehicle estimated position – $f(e,\alpha,\theta)$ has to be chosen, differently from Indiveri (1999), taking into account the vehicle position with respect to the ellipsoidal domain, as it follows

$$f(e,\alpha,\theta) = v_d\left(1 - \frac{V(e,\alpha,\theta)}{\varepsilon}\right)$$

This function automatically decreases the target velocity $\dot{s}$ as the distance of the vehicle with respect to the ellipsoid centre increases.

2.2 *Tuning the controller parameters*

The control law herein presented acts on two variables, the vehicle velocity and the path curvature, and is characterised by five parameters γ, h, β, λ, ε that the user has to tune.
Assuming that only the kinematic model is considered, the curvature is algebraically related to the steering angle, that represents the actual control variable.
There is, however, no straightforward relation between the controller parameters and the behaviour of the closed loop system, making the regulator tuning a rather complex procedure.
In order to support and simplify the selection of these parameters, a procedure to relate their values to the desired trajectory tracking error is here introduced. A further refinement of the controller parameters can be then performed using an accurate simulator of the vehicle, as it will be discussed in Section 3.

Before introducing the parameter selection procedure, it is useful to remember that the tuning should be driven by the following constraints

$$\gamma > 0, \quad h > 1, \quad \lambda > 0$$

and

$$2 < \beta < h+1, \quad 0 < \varepsilon < \frac{\pi^2}{4} \tag{7}$$

From relation (6) it follows that the principal semi-axis of the ellipsoid in the space (e,α,θ), that is used to trigger a parking or a trajectory tracking problem, have the following expressions

$$a = \sqrt{\frac{\varepsilon}{\lambda}}, \quad b = \sqrt{\varepsilon}, \quad c = \sqrt{\frac{\varepsilon}{h}} \tag{8}$$

where a,b,c are the semi-axis along the directions e,α,θ, respectively. Relations (8) allow to formulate an initial guess at least for parameters λ, ε and h. The values of these parameters determine, by way of (8), the characteristics of the ellipsoidal volume V. As a consequence, they have to be chosen in such a way that there is a reasonable neighbourhood around the origin of the space (e,α,θ) where the trajectory tracking controller can operate compensating for trajectory deviations. In other words, the size of the ellipsoidal semi-axis has to be chosen in such a way that when the vehicle is far from its desired pose a parking problem is triggered. On the other side, however, when the vehicle is driving around the desired path one has to avoid that the controller switches too frequently between a parking problem and a trajectory tracking one, causing an oscillating behaviour. This is of particular importance, for example, in order to make the controller insensitive to the localisation error as much as possible.

3. EXPERIMENTAL RESULTS

The vehicle considered in this research is a YAMAHA GRIZZLY 700 (see Table 1 for a list of the main characteristics), a commercial fuel-powered utility ATV, specifically designed for agriculture work (Figure 2). For the purposes of the project, the vehicle has been equipped with three low-level servomechanisms, to automatically regulate the steer position, the throttle aperture and the braking force Bascetta et al. (2009), and with many different sensors, as it will be described in the following.

3.1 *Hardware and software architecture*

In order to allow for teleoperation and autonomous navigation, an on-board hardware/software control architecture has been developed.

Main characteristics of the vehicle	
Engine type	686cc, 4-stroke, liquid-cooled, 4 valves
Drive train	2WD, 4WD, locked 4WD
Transmission	V-belt with all-wheel engine braking
Brakes	dual hydraulic disc (both f/r)
Suspensions	independent double wishbone (both f/r)
Steering System	Ackermann
Dimensions (LxWxH)	2.065 x 1.180 x 1.240 m
Weight	296 Kg (empty tank)

Table 1. ATV characteristics

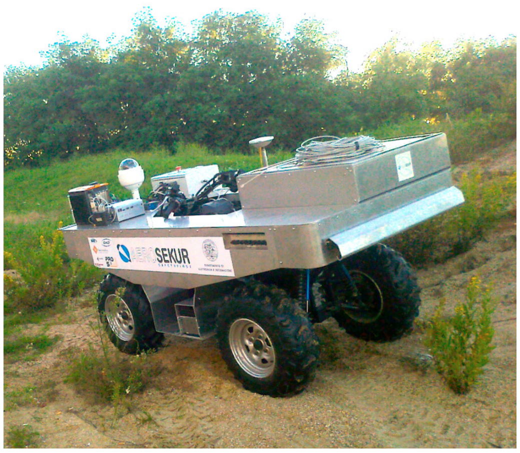
Fig. 2. The ATV used in the experiments.

The architecture is divided into two different layers: the higher level is developed using ROS and is responsible for acquiring data from external sensors, such as GPS, magnetometer, Inertial Measurement Unit (IMU), cameras and laser range-finders (Figure 3). Moreover, it hosts the modules for localisation, path planning, high-level trajectory control and autonomous driving. The lower level acts as an interface between the vehicle servomechanisms and the ROS architecture: it receives desired set points from the higher level, reads the steering angle, throttle aperture, vehicle speed measurements and runs the low-level control loops.

To implement such an architecture that includes high-level and low-level tasks, a multi-layered and multiprocessor hardware/software architecture is required, which consists of: an industrial PLC, which allows a good compromise between the hard real-time requirements and high-level programming, and a standard i5 PC, on which the high-level ROS architecture (perception, localisation, obstacle avoidance, medium-long range navigation, planning, etc.) runs. Communication between the two layers is obtained through an Ethernet link.

The core part of the perception architecture consists in the localisation node, which is based on ROAMFREE Cucci and Matteucci (2014b). This open source framework provides out-of-the-box 6-DOF pose tracking fusing the information coming from an arbitrary number of heterogeneous information sources[2]. Calibration procedures to account for distortions, biases, misalignments between sensors and the main robot reference frame are provided as well. The information fusion problem is formulated as a fixed-lag smoother and it runs in real time thanks to efficient implementations of the inference algorithms (see Cucci and Matteucci (2014a) for further details). At the present stage of development the localisation module estimates the robot poses exploiting vehicle kinematic data (i.e., the steer position and the rear wheel speed), GPS, magnetometer, and the gyroscopes in the inertial measurement unit.

[2] http://roamfree.dei.polimi.it

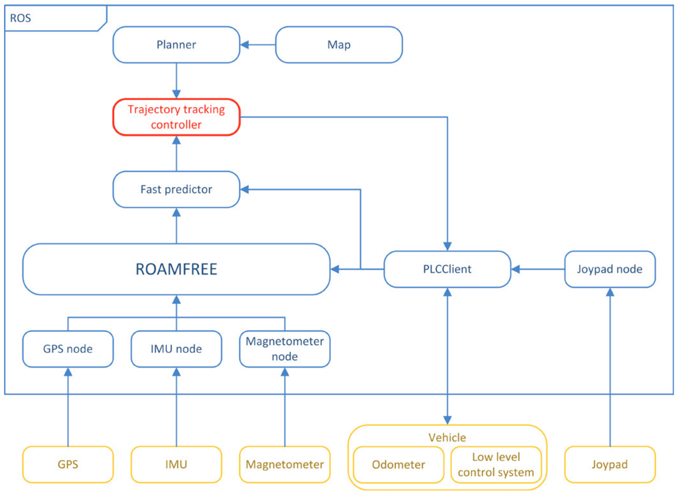

Fig. 3. The real system architecture.

The pose estimate is generated by the localisation node at a frequency of 20 Hz. Then, to prevent the detrimental effect of possible network delays, a predictor node has been introduced. This node computes a prediction of the future robot pose, based on the latest available global pose estimate and on the integration of the Ackermann kinematic model, at a frequency of 50 Hz.

Given a map and a goal, a planner node, based on the SBPL library[3], determines a global path, which is then interpolated with respect to the natural coordinate and fed to the trajectory tracking module. This module computes set points for the vehicle speed and steering angle, based on the current pose and velocity estimates, and on the planned trajectory. These set points are sent to the low-level regulators by a ROS node communicating with the PLC, which additionally acts as a multiplexer between the autonomous drive and the manual set points (teleoperation), depending on the current operating mode.

3.2 *Tuning the controller parameters*

The relations introduced in Section 2 are here exploited to determine the controller parameters, in such a way that the controller specifications are fulfilled and the control system is robust with respect to localisation errors.

Considering the three parameters ε, h and λ, they are related to the size of the ellipsoid in the space (e, α, θ), that is used to trigger a parking or a trajectory tracking problem, by way of relations in (8). Consequently, selecting the size, i.e., the semi-axis, of this ellipsoid, the values of these parameters are determined.

We start assuming that the parking problem is triggered only when the vehicle is very far from the target frame, i.e., if the vehicle starts from a position that is not close to the desired path, or if a deviation from this path is commanded due to the presence of a wide obstacle. A distance of 30 meters is thus

[3] http://wiki.ros.org/sbpl

selected for a, the semi-axis that defines the ellipsoid size in the e direction.
Concerning the orientation error, we assume the parking problem is triggered only when it is more than 90 degrees. If this error is equally divided between α and θ, the two other semi-axis, b and c, are both equal to $\pi/4$.
From relations (8) it now follows

$$\varepsilon = \frac{\pi^2}{16} \approx 0.617, \quad \lambda = \frac{\pi^2}{14400} \approx 6.85 \cdot 10^{-4}, \quad h = 1$$

In order to complete this initial guess a unitary value for the gain γ is selected. The value of β, instead, is chosen equal to 2, in order to satisfy the constraint in (7).

Starting from this preliminary parametrisation the controller can be validated using an accurate vehicle simulator, and a fine tuning of the parameters can be performed. For this purpose, a multi-body model of the vehicle, aiming at representing all the control-relevant dynamics, has been developed using the Modelica language and the Vehicle Dynamics Library Damelio et al. (2015). A thorough validation of this model against experimental data has been performed as well (see Damelio et al. (2015) for further details).
The trajectory that has been used as a benchmark to validate the controller is composed of a left and a right turn with a straight path in the middle. The desired vehicle velocity has been selected as a function of the local path curvature, so as to avoid roll-over phenomena. The same trajectory has been then performed considering turns with different curvature radii, in order to test the robustness of the system.
Considering the two parameters γ and β, whose values have been arbitrarily assigned, β is the more difficult to tune. In fact, γ has a clear interpretation as it represents the proportional gain between the distance error and the vehicle velocity. The simulations are thus mainly focused on determining a suitable value for β. This selection can be performed with the aim of minimising function V along the path, without affecting the controller performance. In fact, having a smaller value of V along the path implies more robustness and, thanks to the chosen radial function $f(e,\alpha,\theta)$, an higher vehicle velocity.

Another important role played by the validation on the simulator is related to the fact that the nonlinear controller here proposed does not directly take into account actuator dynamics and saturation. This aspect is particularly relevant especially for the steering command: the validation allows to select the controller gains in such a way that the finite bandwidth of the steering servo-loop is taken into account.

As a result of the fine tuning, the following controller parameters were selected

$$\gamma = 1, \quad \beta = 2.2, \quad h = 1.5, \quad \lambda = 0.001, \quad \varepsilon = 0.9$$

In order to validate the behaviour of the overall localisation and trajectory tracking systems and check the correctness of the software, the control system running on the vehicle has been further tested using v-rep [4], a simulation environment that can be easily integrated with ROS Bardaro et al. (2014).

3.3 Results

In order to show the effectiveness of the proposed control law, three different experiments have been performed.

[4] `http://www.coppeliarobotics.com/`

In the first experiment an eight-shaped trajectory has been considered (Figure 4). The trajectory originates one meter ahead with respect to the current position of the vehicle and it follows the path starting from the smaller circle and doing a left turn first. The bigger circle has a diameter of 18 meters, while the smaller one has a diameter of 12.5 meters. The vehicle drives along this path at a maximum velocity [5] of 2 m/s.
In this experiment the vehicle position is estimated with a frequency of 20 Hz, i.e. the kinematic predictor is not used.
Figure 4 shows the results of the trajectory tracking experiment. In particular, the black dashed line, the blue solid line and the red crosses represent the reference path, the vehicle position estimated by ROAMFREE, and the raw GPS readings, respectively. Despite GPS inaccuracies, ROAMFREE is able to estimate a reasonable vehicle position, as it can be seen comparing the localisation and GPS raw readings where GPS information is reliable, and to account for compromised sensor readings. Concerning the trajectory following, the picture reveals that the vehicle follows the path with reasonable accuracy, exhibiting under-steering/over-steering especially when the localisation is partially compromised by a faulty GPS behaviour.
This first experiment shows that the overall localisation and control architecture are solid, but, as expected, also that the accuracy and, especially, the update rate of the vehicle position estimation highly affect the performance of the trajectory tracking system.

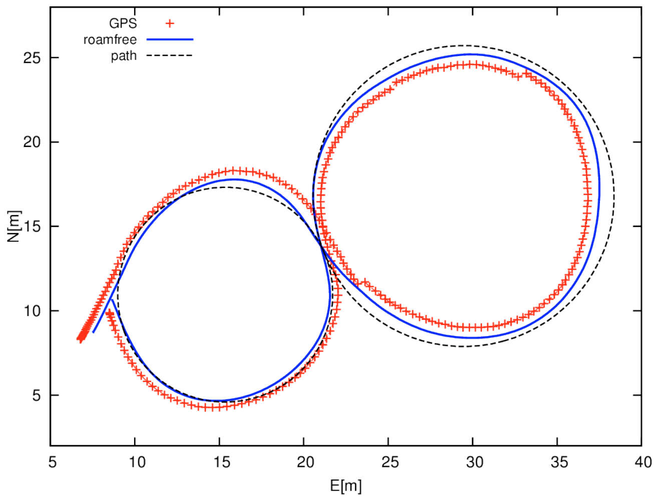

Fig. 4. Autonomous trajectory following: an eight-shaped path

The second experiment concerns a sinusoidal path characterised by an amplitude of 2 meters, and a distance between two peaks of 15 meters (Figure 5). The vehicle drives along this path at a maximum velocity of 3 m/s.
In the third experiment a rectangular-shaped path with smooth turns at the corners is considered (Figure 6). The path includes four left turns and four straight lines, the longest are 15 meters each, the shortest 2 meters. The vehicle repeats the path twice at a maximum velocity of 3 m/s.
In these experiments the vehicle position is estimated with a frequency of 50 Hz, using the kinematic predictor.
Figures 5 and 6 show that, thanks to the increased update rate of the position estimate, the vehicle follows the two trajectories

[5] In all the experiments the desired velocity is computed limiting the maximum velocity with a function of the path curvature in order to limit the roll-over risk along sharp curves.

more closely. The over-steering effect, though less evident, is still present. As it can be noticed from Figure 6, however, the over-steering behaviour occurs only when the vehicle enters a curve and it is caused by a velocity that is higher than the desired one. This behaviour is due to the fact that, in the actual setup, the brake is not used and the vehicle speed can be reduced only closing the throttle and exploiting the engine braking effect.

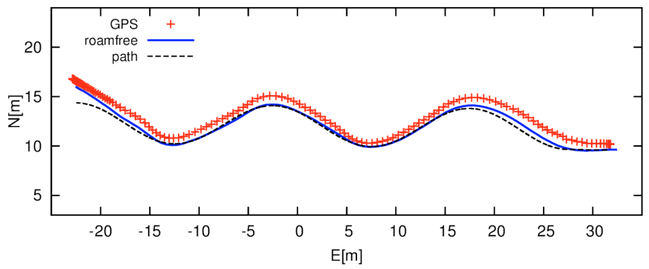

Fig. 5. Autonomous trajectory following: a sinusoidal path.

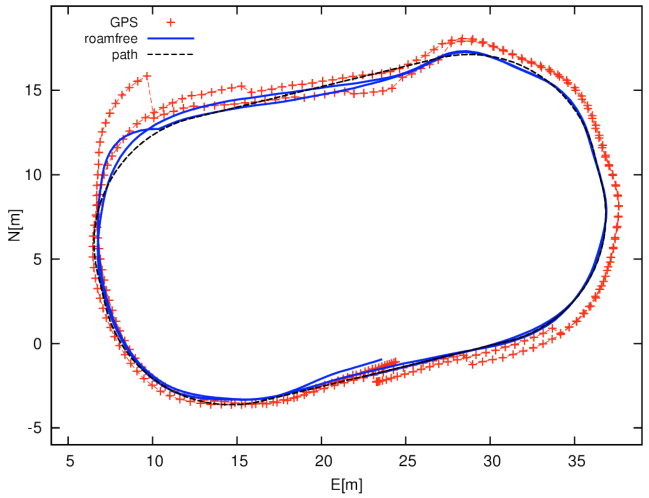

Fig. 6. Autonomous trajectory following: a rectangular-shaped path.

In conclusion, the previous results show the effectiveness of the proposed control law, even in the presence of localisation inaccuracies. The performance of the trajectory tracking controller can be quantitatively measured computing the Hausdorff distance between the desired and executed paths. These distances are 0.897 and 0.545 meter, for the eight-shaped and the sinusoidal paths, respectively. As for the rectangular-shaped path, the distance is 0.505 meter for the first repetition and 1.0 meter for the second one.

The paper is also accompanied by a video attachment[6] showing the results of the last two experiments. The video shows, in addition to some shots taken during the execution of the trajectory tracking experiment, the rviz[7] visualisation of the desired path (green solid line), the raw GPS readings (white dots), the vehicle pose estimated by ROAMFREE (blue solid line and red/green coordinate frame), and the position of the target frame (red arrow).

[6] `http://http://goo.gl/ryHBND`
[7] `http://wiki.ros.org/rviz`

4. CONCLUSIONS

This paper presents a novel trajectory tracking controller for an Ackermann steering vehicle, whose main characteristics are the simplicity of the implementation, the robustness with respect to localisation errors, the possibility to automatically managed a parking and a trajectory tracking problem ensuring smooth variations in the control signals.
A thorough experimental validation of the overall control system, including the trajectory tracking and the localisation subsystems, has been also presented showing the effectiveness and limitations of the proposal.

REFERENCES

Aicardi, M., Casalino, G., Bicchi, A., and Balestrino, A. (1995). Closed loop steering of unicycle like vehicles via Lyapunov techniques. *IEEE Robotics and Automation Magazine*, 2(1), 27–35.

Ballucchi, A., Bicchi, A., Casalino, A., and Balestrino, G. (1996). Path tracking control for Dubin's car. In *IEEE International Conference on Robotics and Automation*, volume 3, 3123–3128.

Bardaro, G., Cucci, D.A., Bascetta, L., and Matteucci, M. (2014). A simulation based architecture for the development of an autonomous all terrain vehicle. In *International conference on Simulation, Modeling, and Programming for Autonomous Robots*, 74–85.

Bascetta, L., Magnani, G., Rocco, P., and Zanchettin, A. (2009). Design and implementation of the low-level control system of an all-terrain mobile robot. In *International Conference on Advanced Robotics*, 1–6.

Cucci, D.A. and Matteucci, M. (2014a). On the development of a generic multi-sensor fusion framework for robust odometry estimation. *Journal of Software Engineering for Robotics*, 5(1), 48–62.

Cucci, D.A. and Matteucci, M. (2014b). Position tracking and sensors self-calibration in autonomous mobile robots by gauss-newton optimization. In *IEEE International Conference on Robotics and Automation*, 1269–1275.

Damelio, E.L., Bascetta, L., Cucci, D.A., Matteucci, M., and Bardaro, G. (2015). A modelica simulator to support the development of the control system of an autonomous all-terrain mobile robot. In *Vienna Conference on Mathematical Modelling*.

Indiveri, G. (1999). Kinematic time-invariant control of a 2D nonholonomic vehicle. In *IEEE Conference on Decision and Control*, volume 3, 2112–2117.

Kim, Y. and Minor, M. (2005). Bounded smooth time invariant motion control of unicycle kinematic models. In *IEEE International Conference on Robotics and Automation*, 3676–3681.

Kim, Y. and Minor, M. (2008). Kinematic motion control of wheeled mobile robots considering curvature constraints. In *IEEE International Conference on Robotics and Automation*, 2527–2532.

Morro, A., Sgorbissa, A., and Zaccaria, R. (2011). Path following for unicycle robots with an arbitrary path curvature. *IEEE Transactions on Robotics*, 27(5), 1016–1023.

Sgorbissa, A. and Zaccaria, R. (2009). A minimalist feedback control for path tracking in cartesian space. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2952–2957.