INTEGRATED PATH PLANNING AND TRACKING FOR AUTONOMOUS CAR-LIKE VEHICLES MANEUVERING

Fernando Gómez-Bravo, Diego A. López
Departamento de Ingeniería Electrónica, S. I. y Automática, University of Huelva, 21819 La Rábida, Huelva, Spain

Francisco Real, Luis Merino, José M. Matamoros
Robotics, Vision and Control Group,University of Seville, Camino de los Descubrimientos, 41092 Sevilla, Spain

Keywords: Autonomous Car-like vehicle, Maneuvering, Motion Planning, Path-tracking.

Abstract: This paper proposes a new method for control car-like vehicles maneuvering. For this purpose, traditional planning and tracking algorithms has been modified in order to follow complex maneuvers in a frame of a distributed control architecture. Thus, planning and tracking algorithm run in different platforms exchanging data in order to control the maneuvers performance.

1 INTRODUCTION

Autonomous navigation of car-like vehicles is a well known topic that has attracted the attention of many researchers (Paromtchik et al., 1998), (Ollero et al., 1999), (Gomez-Bravo et al., 2001), (Wada et al., 2003), (Cuesta et al., 2004), (Daily and Bevly, 2004). Navigation in cluttered scenarios usually involves maneuvering that require stopping the vehicle and changing the sign of the vehicle velocity to avoid collisions. The non-holonomic constraints of the car-like vehicles play an important role. Path tracking (Ollero et al., 1994), (Ollero et al., 1996) and path planning techniques (Latombe, 1991), (Laumont et al., 1994), (LaValle, 2006) have been largely approached in mobile robot literature. However, generating and tracking car-like maneuvers are not frequently reported (Cheng et al., 2001), (Wada et al., 2003), (Cuesta et al., 2004). This paper presents a new integrated architecture particularly designed for planning and tracking maneuvers. Furthermore, this strategy can be also applied when complex maneuvers are not required.

Real-time path planning usually requires significant computational resources that could overload the limited on-board processing capability. Then, a suitable alternative is to implement the planner on external dedicated servers that could provide this capability to one or several networked autonomous vehicles. These servers could be also networked with sensors in the environment providing information for navigation (Gomez-Bravo et al., 2007).

In this paper a distributed implementation integrating both maneuvers planning and tracking techniques for autonomous car-like maneuvering is presented. The novelty of this method is the adaptation of the planning and tracking method so that both can work in different computer, establishing a communication process between the planner and the tracker system in order to control effectively the maneuver performance.

On the one hand, the planning method applied in this paper is based on the Rapidly Exploring Random Trees algorithm (RRT) (LaValle, 1998), (Bruce and Veloso, 2002) (LaValle, 2006). This technique can be easily extended to non-holonomic vehicles providing simple solutions for car-like maneuvers generation (LaValle and Kuffner, 1999), (Gomez-Bravo et al., 2008). In the present approach, besides obtaining path for maneuvering in complex scenarios, the planner is capable of dividing the originally computed maneuver into different sections in order to facilitate the tracking task and allowing to modify easily the original path if changes in the initial map are detected. On the other hand, the path tracking technique is a modified version of the well known pure pursuit geometric approach (Ollero, 2001). The original tracking strategy has been modified so that this method can be applied for maneuvers tracking. Moreover, and different from previous approaches, in the

Gómez-Bravo F., A. López D., Real F., Merino L. and M. Matamoros J. (2009).
INTEGRATED PATH PLANNING AND TRACKING FOR AUTONOMOUS CAR-LIKE VEHICLES MANEUVERING.
In *Proceedings of the 6th International Conference on Informatics in Control, Automation and Robotics - Robotics and Automation*, pages 457-464
DOI: 10.5220/0002219504570464

architecture presented in this paper the planner determines the maneuver performance by establishing some of the tracker parameters.

The paper is organized as follows: in the next section the basis of the car-like vehicles maneuvering is introduced and the application of the RRT to maneuvers generation is also presented. Section 3 is devoted to describe the adaptation of the path tracking algorithm for maneuvering. In section 4 the proposed implementation is presented. Finally, in section 5, experimental results, obtained with a real autonomous car-like vehicle when maneuvering in an outdoor scenario, are presented. The paper ends with the conclusions and the references.

2 CAR-LIKE VEHICLES MANEUVERING

2.1 Car-Like Vehicles

Car-Like vehicles are non-holonomic systems characterized by kinematics constraints resulting in nonintegrable differential equations that should be taken into account for motion planning and obstacle avoidance.

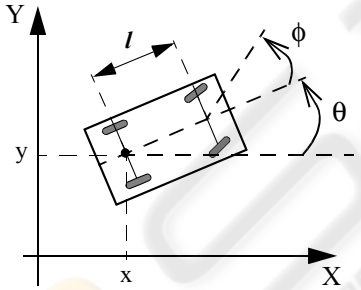

Figure 1: Car-like vehicle.

Usually, the kinematics model of car-like vehicles is expressed as:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} v(t) \\ v(t)\frac{\tan\phi(t)}{l} \end{bmatrix} \quad (1)$$

where (x,y) is the position of the rear reference point, and θ is the vehicle's heading, both in a global reference frame, $v(t)$ is the linear velocity, $\phi(t)$ is the steering angle, that defines the curvature of the path, and l is the distance between the rear and front wheels (see Fig. 1).

Typically, the values of $\phi(t)$ are constrained by the value ϕ_{max}, accomplishing the relation

$$R_{min} = \frac{l}{\tan\phi_{max}} \quad (2)$$

where R_{min} is the minimum curvature radius.

Each wheel of the vehicle presents a kinematics constraint which prevents the robot from moving to the orthogonal direction of the wheels longitudinal axe. Then, the kinematics of these vehicles is usually characterized by the following differential non-holonomic equation

$$\dot{x}\cdot\sin\theta - \dot{y}\cdot\cos\theta = 0. \quad (3)$$

In the next section, the basis of the method for coping with this constraint, when planning trajectories, are presented.

2.2 Continuous Navigation vs. Maneuvering

There are different approaches to car-like vehicles motion planning (Laumont et al., 1994), (Cheng et al., 2001), (Gomez-Bravo et al., 2008). Path planning techniques provides trajectories based on different searching techniques (Latombe, 1991), (LaValle, 2006). Particularly, the randomized methods have attracted considerable attention (Barraquand and Latombe, 1991), (Amato and Wu, 1996), (Cheng et al., 2001), (Bruce and Veloso, 2002). Due to the non-holonomic nature of these vehicles, many of these methods require further processing in order to obtain a path accomplishing the kinematics constraints. Thus, derivable and continuous curvature trajectories are frequently provided by these techniques. However, when cluttered scenarios are involved, complex maneuvers may be needed, and path generation should also include *inversor configurations*, (Latombe, 1991), i.e configurations where the sign of the vehicle velocity changes. In these cases, discontinuous curvature trajectories can also be considered as admissible paths, although the kinematics constraints still have to be accomplished.

2.3 Planning Maneuvers

General strategies for maneuvers generation have not been frequently addressed, (Latombe, 1991), (Laumont et al., 1994), (LaValle, 2006). Recent approaches have focused attention in car-like parking maneuvers (Paromtchik et al., 1998), (Gomez-Bravo et al., 2001), (Cuesta et al., 2004).

The method presented in this paper is based on a planner previously presented in (Gomez-Bravo

et al., 2007) and (Gomez-Bravo et al., 2008). This method takes the advantage of a randomized generation process, the original Bidirectional RRT algorithm (LaValle, 1998), (Bruce and Veloso, 2002) adapted to non-holonomic vehicles. Firstly, RRT is used without considering any kinematics constraints providing a data structure (two trees with free collision configurations) connecting the initial with the goal configuration (see Fig. 2).

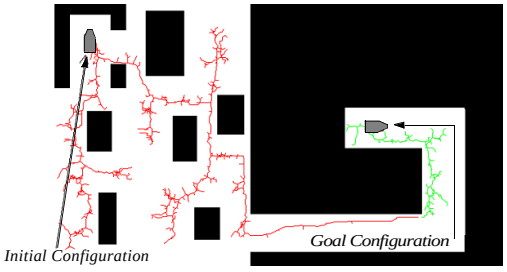

Figure 2: RRT connecting trees.

In the next step, the tree provided by the algorithm is turned into a sequence of feasible admissible paths by means of a set of *connecting path* previously designed (see Fig. 3). Each of these connecting paths are built from a set of basic canonical maneuvers. Examples of these canonical maneuvers for car-like vehicles have been previously reported in (Gomez-Bravo et al., 2001), (Cuesta et al., 2004) and (Gomez-Bravo et al., 2008).

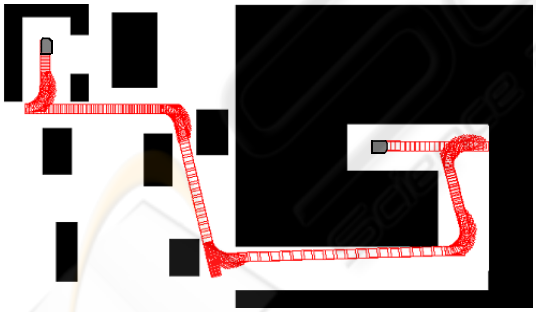

Figure 3: Final path.

As it is shown in (Gomez-Bravo et al., 2008), by means of this procedure, any two configurations can be connected by an admissible path. It is remarkable that, with this methods, the planner generates trajectories for both strategies: traditional continuous navigation, without stopping, or maneuvering navigation when necessary. That is, the planner includes inversors on the trajectory when a cluttered environment requires the vehicle to stop several times.

3 MANEUVERING NAVIGATION CONTROL

In this section the basis of the tracking method are illustrated. The particular problems of performing maneuvers are also addressed. Finally, the convenient modifications of the path following method for executing complex maneuvers are presented.

3.1 Continuous Path-tracking

Different approaches have been proposed for path following, where the vehicle position is estimated by an Extended Kalman Filter (EKF) (Grewal and Andrews, 1993) using Global Positioning System (GPS) (Daily and Bevly, 2004) and odometry.

In the approach presented in this paper, an accurate path-tracking strategy is accomplished by applying a modified version of the "Pure-pursuit" algorithm (Ollero, 2001). In this algorithm a point of the reference path is selected at each time instant so that the steering angle is obtained according to the expression:

$$\phi = \arctan(l \cdot \frac{2\triangle}{L_H^2}) \qquad (4)$$

where l is defined in Fig. 1 and L_H is a parameter called the look-ahead, which represents the distance between the target point and the vehicle's current position, and $\triangle$ is the lateral distance between the robot and the target point (see Fig. 4).

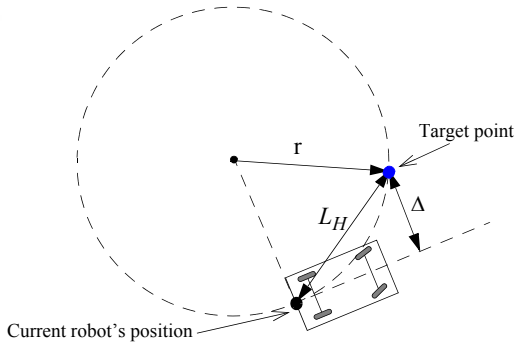

Figure 4: Circular trajectory.

With this driving strategy the vehicle will follow a circular arc from its current position to the target point. Usually the point of the path is selected by the following procedure: a) firstly, the nearest point, (x_n, y_n), to the vehicle is obtained; b) secondly, from (x_n, y_n) the target point (x_t, y_t) is found as the one which is at the distance L_H, from the current vehicle's position in the direction of the desired motion, (see Fig. 5). In

this way at each time instant, from an estimated vehicle configuration, a new point of the path is selected and the vehicle navigates following the desired direction.

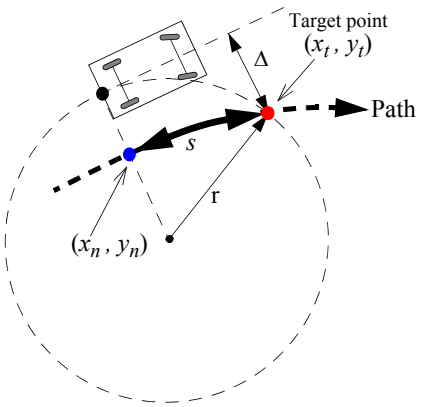

Figure 5: Pure-pursuit.

There could be different criteria for selecting the value of L_H according to the velocity and the shape of the path. In this approach the selection of the velocity and the look-ahead value is determined by the planner taking into account the characteristics of the planned trajectory.

3.2 Maneuvers Path-tracking

Traditionally, continuous navigation is based on a sequential procedure: first the planner provides a path connecting the starting point with the goal configuration and finally the tracking algorithm is applied so that the vehicle follows the trajectory until the vehicle arrives to the end of the path. In this way, path-following algorithms are usually designed so that the tracking error decrease as the vehicle follows the reference path. However, errors use to increase when the vehicle arrive to the goal configuration. Clearly, stopping the vehicle involves slipping phenomenon, particularly when the vehicle navigates outdoor on irregular terrains. Obviously, this is a drawback, specially when the vehicle is performing maneuvers as the vehicle has to stop several times. Thus, if a trajectory to be followed presents inversors, the traditional procedure need to be improved.

The present approach is based on developing two independent modules, the planner and the tracker, that take the responsibility of planning and tracking the maneuver respectively. These two modules will interact so that they manage the maneuver performance. The planner, once a initial trajectory has been generated, will divide it in *path sections*, separated by inversors. Each path section will have associated a kinematics profile and a look-ahead determined by the planner. The tracker will receive sequentially each of the path sections and will apply the tracking algorithm until the inversor configuration is reached. When the vehicle is stopped over a inversor the tracker will ask the planner for the next section of the path. This procedure will be repeated until the whole maneuver is performed.

Clearly, stopping the vehicle over the inversor is a very difficult task. A simple approach for managing the stopping process consist on decreasing, the vehicle velocity as it is closed to the inversor. The tracker will stop the vehicle when the position error is lower than a threshold. However a problem could appear when this strategy is applied. If the vehicle configuration and the inversor fall in a circular arc which radius is shorter than minimum curvature radius, R_{min}, the vehicle will be trapped in a circular trajectory in which the position error never decrease (see Fig. 6).

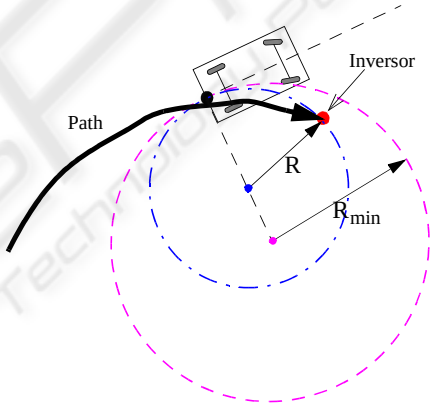

Figure 6: Pure pursuit problem.

Thus, the tracking algorithm will stop following that section of the path if this situation is detected, and asks the planner for the next section to follow.

4 DISTRIBUTED IMPLEMENTATION

Motion control of autonomous vehicles requires real time attention to different task. For instance, getting the GPS lecture, reading the proximity sensors, running the path tracking algorithm etc, are procedures that need a tailored use of the time into the control loop. Nevertheless, path planning usually presents a high computational cost. Then, implementing these algorithms and the vehicle control program into the same computer machine could negatively affect the

distribution of the time control loop. In order to avoid this problem, in this approach, the planning algorithm is executed on a computer different to the one containing the autonomous vehicle control program. Thus a distributed strategy has been implemented by using an auxiliary server. This computer has been configured so that planning task are executed. Moreover other external interactions are also performed (collecting information from external sensors or attending human interface for instance). The novelty of this approach is based on the relation between the vehicle control program and the services running into the auxiliary server. As was mentioned in the previous section, there are parameters of the tracking algorithm (velocity and look-ahead) that are computed in the auxiliary server from the path provided by the planning algorithm.

The proposed implementation is shown in Fig. 7. The auxiliary server communicates with the autonomous vehicle by means of WIFI devices, using the TCP protocol. In this way the YARP protocols and services have been used. Considering the OSI Reference Model the YARP protocols could be placed on the Session Layer. This procedure allow transmitting object data defined in the C/C++ format. These protocols provide several net ports independent from the operative system. Thus, it represents a flexible solution for implementing the dialog between the navigation control algorithm and the process running at the auxiliary server, being independent from the computer platform.

The global planner program has two process running in parallel. The first one is the planner itself, i.e, the responsible of providing admissible paths. The second one, establishes and maintains the communication between the planner and the tracker through the YARP port. By means of this process the planner receives continuously the vehicle position and the goal point over the current path section. If the vehicle navigates too far from reference path or the scenario changes, the planner decides whether the current path section is computed again or not. As was described before, the path is divided into sections separated by the inversor configurations, and the vehicle follows sequentially the path sections. Thus, in case of any possible eventuality (new obstacles on the map, large position error or the vehicle being trapped by a circular trajectory), the planner modifies the affected path section. Only if these changes involve the current path section, the vehicle has to stop. Otherwise, the modifications are transmitted to the autonomous vehicle in a sequential procedure.

Additionally, the planner determines the value of the look-ahead and velocity associated to each section of the path. It establishes these values according to their characteristic, taking into account the curvature and the length. An heuristic procedure for setting the vehicle velocity has been implemented. Thus a long and straight path section can be followed with a high linear velocity whereas a short and curved section requires a low velocity value. Likewise, different criteria for selecting the Look-ahead according to the velocity and the shape of the path can be found in (Ollero et al., 1994) and (Ollero et al., 1996).

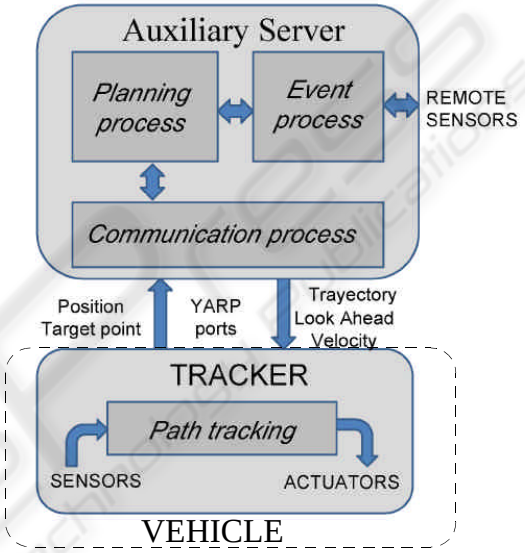

Figure 7: Distributed implementation.

5 EXPERIMENTAL RESULTS

The experiments presented in this section have been performed with ROMEO-4R, an autonomous car-like vehicle build at the Sevilla University (Ollero et al., 1999). The navigation control program runs into an industrial PC-Pentium III under Linux (Debian 2.6). The control program has been implemented in C++ and attends different concurrent process. Thus an EKF has been implemented so that GPS and odometry data are combined to obtain a robust pose estimation. At the same time, the path tracking algorithm is executed, applying the techniques described above, being continuously connected whit the planner program at the auxiliary server. Likewise, the autonomous vehicle also attends different type of proximity sensor (ultrasonics, LIDAR etc) that can be used for the improvement of the local obstacle avoidance. This technique is not addressed in the paper. The auxiliary server is implemented in a laptop computer, with a 1.8 Ghz AMD processor under Windows XP.

The planner and all the services running in the auxiliary server have been implemented in Java. Additionally, a connection with an external wireless sensor network has been also developed (Gomez-Bravo et al., 2007).

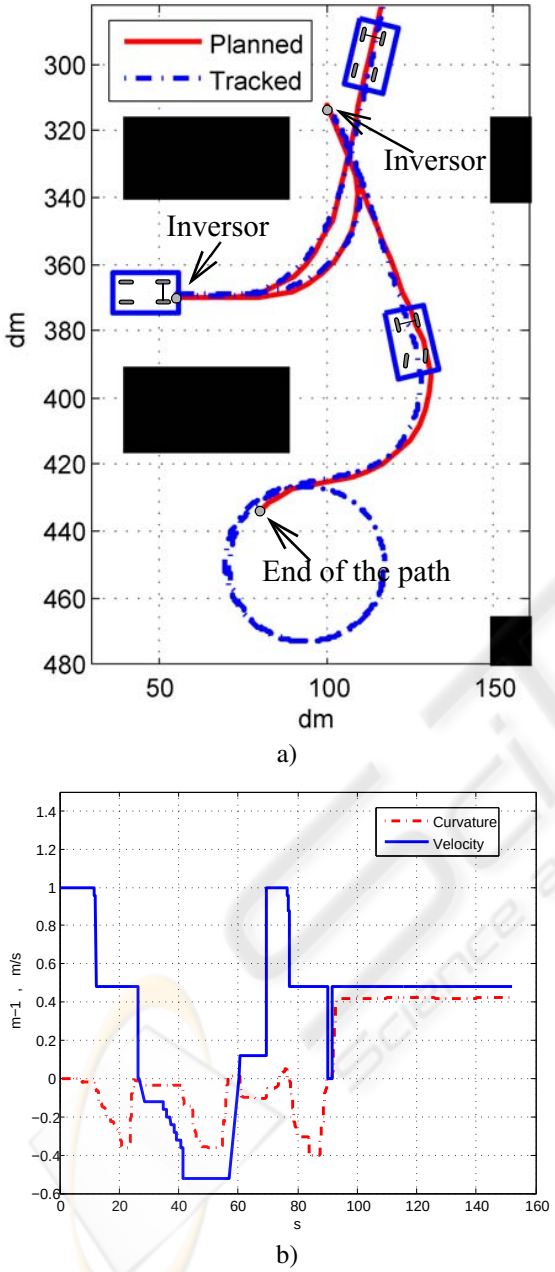

Figure 8: Experiment 1: a) Vehicle trajectory and b) Vehicle velocity and curvature.

Fig. 8 a) shows an experiment in which the path presents three sections and two inversors. In this experiment the vehicle fell trapped at the end of the path, it was specially configured in other to illustrate this type of situation. Fig. 8 b) presents the evolution of the velocity and curvature. Note that the velocity kept constant value as the vehicle navigated around the inversor and the curvature got the maximum value that represents the minimum curvature radius.

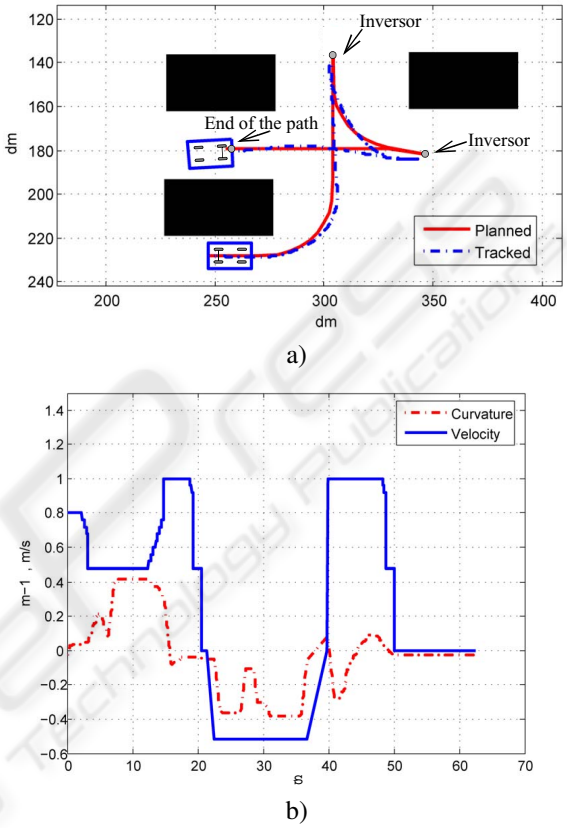

Figure 9: Experiment 2: a) Vehicle trajectory and b) Vehicle velocity and curvature.

In Fig. 9 a) a new experiment is shown. Due to the irregularities of the terrain, the vehicle motion is affected by perturbations that make the vehicle trajectory being different to the reference path. However, the vehicle finally achieved the final desired configuration. Fig. 9 b) presents the evolution of the velocity and the curvature. Observe how, velocity was decreased along the curved section and was increased when a straight section was followed.

Finally in Fig. 10 some pictures of ROMEO 4R performing a maneuver are shown. These pictures were recorded during the experiment presented in Fig. 9

Figure 10: ROMEO 4R performing the second maneuver.

6 CONCLUSIONS

This paper presents a new approach for autonomous car-like vehicles maneuvering. The method allows navigation of robots with non-holonomic constraints in cluttered scenarios, providing, when necessary, continuous paths or complex maneuvers, where the vehicle has to change the sign of the velocity. This approach allows to distribute the computational task for computing the path and tracking the maneuver. Thus, a new implementation has been proposed so that the planning and the tracking algorithm exchange continuously data in order to enhance the maneuver performance. The method has been validated in real experiment with the autonomous car-like vehicle ROMEO-4R built at the Sevilla University.

ACKNOWLEDGEMENTS

This work has been supported by the National Spanish Research Program, project DPI2008-03847, and the Project URUS funded by the European Commission under grant IST-045062.

REFERENCES

Amato, N. M. and Wu, Y. (1996). A randomized roadmap method for path and manipulation planning. *IEEE Int. Conf. Robot. and Autom*, pages 113–120.

Barraquand, J. and Latombe, J. C. (1991). Robot motion planning: A distributed representation approach. *Int. J. Robot*, pages 167–194.

Bruce, J. and Veloso, M. (2002). Real-time randomized path planning for robot navigation. *International Conference on Intelligent Robots and Systems*, pages 2383–2388.

Cheng, P., Shen, Z., and LaValle, S. M. (2001). Rrt-based trajectory design for autonomous automobiles and spacecraft. *Archives of Control Sciences*, pages 167–194.

Cuesta, F., Gomez-Bravo, F., and Ollero, A. (2004). Parking manoeuvres of industrial-like electrical vehicles with and without trailer. *IEEE Trans. on Industrial Electronics*, pages 257–269.

Daily, R. and Bevly, D. (2004). The use of gps for vehicle stability control. *IEEE Trans. on Ind. Electron*, pages 270–277.

Gomez-Bravo, F., Cuesta, F., and Ollero, A. (2001). Parallel and diagonal parking in nonholonomic autonomous vehicles. *Engineering Aplication of Artificial Inteligence*, pages 419–434.

Gomez-Bravo, F., Ollero, A., Cuesta, F., and Lopez, D. (2007). Rrt-d: A motion planning approach for autonomous vehicles based on wireless sensor network information. *6th IFAC Symposium on Intelligent AutonomousVehicles*, page C.D.

Gomez-Bravo, F., Ollero, A., Cuesta, F., and Lopez, D. (2008). A new approach for car-like robots manoeuvering based on rrt. *Robotica*, pages 10–14.

Grewal, M. S. and Andrews, A. P. (1993). *Kalman Filtering Theory and Practice*. Prentice-Hall.

Latombe, J. C. (1991). *Robot Motion Planning*. Kluwer Academic Pulisher.

Laumont, J., Jacobs, P., Taix, M., and Murray, M. (1994). A motion planner for nonholonomic mobile robots. *IEEE Trans. on Robotics and Autom*, pages 577–593.

LaValle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning. In *TR 98-11*.

LaValle, S. M. (2006). *Planning algorithms*. Cambridge University Press.

LaValle, S. M. and Kuffner, J. J. (1999). Randomized kinodynamic planning. *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 473–479.

Ollero, A. (2001). *Manipuladores y Robots Moviles*. Marcombo Boixareu.

Ollero, A., Arrue, B. C., Ferruz, J., Heredia, G., Cuesta, F., Lopez-Pichaco, F., and Nogales, C. (1999). Control and perception componenets for autonomous vehicle guidance. application to the romeo vehicles. *Control Eng. Practice*, pages 1291–1299.

Ollero, A., Garcia-Cerezo, A., and Martinez, J. (1994). Fuzzy supervisory path tacking of autonomous vehicles. *Control Engineering Practice*, pages 313–319.

Ollero, A., Garcia-Cerezo, A., and Martinez, J. (1996). Design of a robust high performance fuzzy path tracker for autonomous vehicles. *Journal of Systems Science*, pages 799–806.

Paromtchik, I. E., Damm, M., and Matioukhina, L. I. (1998). Autonomous maneuvers of a nonholonomic vehicle. *Intelligent Autonomous Systems. International Scientific Issue*, pages 38–45.

Wada, M., Yoon, K. S., and Hashimoto, H. (2003). Development of advanced parking assistance system. *IEEE Trans. on Ind. Electron*, pages 4–17.