Safe and Reliable Path Planning for the Autonomous Vehicle Verdino

Rafael Arnay, Néstor Morales, Antonio Morell, Javier Hernández-Aceituno, Daniel Perea Ström, Jonay Toledo, Alberto Hamilton, Javier Sánchez-Medina, and Leopoldo Acosta

Universidad de La Laguna, Canary Islands, Spain Email: rafa@isaatc.ull.es

***Abstract*—This paper introduces a local planner which computes a set of commands, allowing an autonomous vehicle to follow a given trajectory. To do so, the platform relies on a localization system, a map and a cost map which represents the obstacles in the environment. The presented method computes a set of tentative trajectories, using a schema based on a Frenét frame obtained from the global planner. These trajectories are then scored using a linear combination of weighted cost functions. In the presented approach, new weights are introduced in order to satisfy the specificities of our autonomous platform, Verdino. A study on the influence of the defined weights in the final behavior of the vehicle is introduced. From these tests, several configurations have been chosen and ranked according to two different proposed behaviors. The method has been tested both in simulation and in real conditions.**

Digital Object Identifier 10.1109/MITS.2015.2504393
Date of publication: 20 April 2016

I. Introduction

Generally speaking, the navigation system of autonomous vehicles is composed of two main levels: the global planner and the local planner. The first level consists of the generation of a feasible route from the current position of the vehicle to a desired goal. The second level computes the necessary commands to control the vehicle in order to follow the global plan, while dynamically adapting to the changing environment conditions.

The problem we want to solve is safely following a predefined route while avoiding dynamic obstacles. This problem is not trivial, since several factors have to be taken into account. For starters, the safety of pedestrians is crucial: the vehicle has to navigate close to the desired route while keeping a safe distance to the obstacles. Secondly, the navigation has to be comfortable from the passengers point of view: when following the global path and avoiding obstacles, the performed maneuvers have to prevent both abrupt changes of linear and angular speed and high curvature trajectories.

IMAGE COURTESY OF HTTP://VERDINO.WEBS.ULL.ES

The method presented in this paper is the local planner of an autonomous robotic prototype called Verdino[1] [1], shown in Fig. 1. This electric vehicle was designed for people transportation in pedestrian environments.

When a new destination is selected, the global planner builds a feasible path from the current vehicle position to the desired goal. The vehicle then follows this path by using the method presented in this paper. In this method, the euclidean space surrounding the vehicle is transformed to the Frenét space, using the computed global trajectory as basis. Then, a set of tentative paths is computed, considering the following restrictions: all paths should start at the position and orientation of the vehicle, and should end parallel to the global trajectory, at a parameterized lateral distance from it. This way paths are computed using just geometrical information, making the problem simpler by not needing to define a kinematic model of the vehicle to generate the trajectories in the euclidean space. Once trajectories are computed, they are transformed back to the euclidean space, in which they are scored based on different variables like, for example, their curvature or their distance to obstacles. Impossible paths (those which can not be followed by the vehicle) are removed. Using these scores, a winner path is selected and used for the computation of the next speed and steering commands.

This vehicle used as a testing platform is a standard golf car, which has been electronically and mechanically modified so it can be controlled by an on-board computer. It is equipped by default with six 6 V batteries, a speed controller, a 36 Vcc electrical motor, mechanical brakes and steering, and has a maximum speed between 19 and 23 Km/h.

In order to localize itself, the vehicle is equipped with an odometry system attached to each wheel, which allows making relative position estimations. This information is combined with the information provided by an Inertial Measurement Unit (IMU) and a centimetric DGPS. Several Light-Detection And Ranging (LIDAR) sensors are used both for SLAM (Simultaneous Localization and Mapping) and obstacle detection. All this information is combined using the method in [2], so the vehicle is properly localized. The vehicle is controlled by an on board computer.

II. Previous Work

In the literature, it is possible to find several planning methods that have been applied in the generation and selection of local paths. Most of these methods are based on a discrete optimization scheme [3], [4], [5] and [6]. From all the approaches of this kind, Rapidly-exploring Random Trees (RRT) and its variants are widely used in non-holonomic motion planning applications. However, real time implementations require efficient heuristics for the sampling configuration. Some examples of this kind of methods are [7], [8] and [9].

[1]http://verdino.webs.ull.es

FIG 1 Verdino prototype.

Some other methods, like the method introduced in this paper, are based in the transformation of the configuration space through the Frenét space. Some examples of this technique are the methods in [5], [3] and [10]. In [5], long term objectives are pursued, like speed keeping, merging, following, stopping. This is done through optimal control strategies within the Frenét frame of the street. In [3], lateral offset is defined as the perpendicular direction to an established base trajectory. This allows the vehicle to drive along the road, parallel to this trajectory. In [10], a set of candidate paths are also generated, with endpoints in fixed positions at different offsets with respect to the base frame, but they do not set this base frame in the center of the road, using a security cost for each candidate path instead. The safety of the path is computed by blurring the binary data of the obstacles.

III. Method

In order for the vehicle to navigate, a global plan has to be defined. This global plan is a rough estimate of the path that the vehicle has to follow to go from its current position to a desired goal.

The global path is generated using the NavFn global planner[2]. This planner implements Dijkstra's algorithm to find the best path through a cost map, which represents the goodness of the navigable areas taking into account static obstacles.

The vehicle can follow the global plan using the local planner. Obstacles are represented in a *costmap*, which is an occupancy grid which the local planner needs in order to select the best trajectories and avoid obstacles.

A. Generation of the Costmap

The costmap maintains information about occupied/free areas in the map, as an occupancy grid. It uses sensor data and information from the static map to store and update information about obstacles in the world, which are marked in the map (or cleared, if they are no longer there).

[2]http://wiki.ros.org/navfn

Each cell in the map can have 255 different cost values:

- A value of 255 means that there is no information about this specific cell in the map.
- 254 means that a sensor has marked this specific cell as occupied. This is considered as a lethal cell, so the vehicle should never enter there.
- The rest of cells are considered as free, but with different cost levels depending on an inflation method relative to the size of the vehicle and its distance to the obstacle.

The cost value of free cells decreases with the distance to the nearest occupied one, following the expression:

$$C(i, j) = \exp(-1.0 \cdot \alpha \cdot (\| c_{ij} - \vec{o} \| - \rho_{\text{inscribed}})) \cdot 253 \qquad (1)$$

In this expression, α is a scaling factor which increases or decreases the decay rate of the cost of the obstacle. $\| c_{ij} - \vec{o} \|$ is the distance between cell $c_{ij} \in C$ (where C is the set of cells in the costmap) and the obstacle. Finally, $\rho_{\text{inscribed}}$ is the inscribed radius, which is the inner circle of the limits of the car. For a better explanation of the way in which the costmap is computed, please refer to [11]. An implementation of this method is available at http://wiki.ros.org/costmap_2d, as part of the Robotic Operating System (ROS) framework used for the development and testing of our approach. In the tests described in section IV a value of $\alpha = 3.0$ has been used.

B. Local Planner

Once the global path is defined, a method to compute the steering and speed commands is needed, in order to control the vehicle along this path. This method should also be able to avoid the obstacles present in the road. This has to be done in a safe and efficient way. The method developed to solve this problem is based on the solution described in [10], in combination with some ideas proposed in [3], taking into account the characteristics of the *Verdino* prototype.

The basic idea of the local path generation is to define a set of *feasible paths* and choose the best option in terms of their cost. The winner path defines the steering and speed commands that the vehicle will use. Having options among local paths is useful to overcome unforeseen obstacles in the road.

The current euclidean coordinate system is transformed into a new system based on the Frenét space. This space is computed as follows: the global path is considered as the base frame of a curvilinear coordinate system. The feasible local paths are defined in terms of this base frame in the following way:

- The nearest point of the main trajectory to the vehicle (where the distance is computed perpendicular to the global path), will be the origin of the curvilinear coordinate system.

- The horizontal axis will be represented by the distance over the global path, along its direction.
- The vertical axis is represented by the vector which is perpendicular to the origin point and points left of the path direction.

In this schema, trajectories can be easily computed in the curvilinear space (that is, maneuvering information is generated). These are then transformed to the original euclidean space, in which the obstacles information is added by assigning costs to each path.

As seen, the local path generation can be divided into two stages: the *candidate paths generation* and *the winner path selection*.

1) Candidate Paths Generation

In this stage, the base frame of the curvilinear coordinate system is defined, so that the algorithm will be able to compute the trajectories in this space as if the global plan were a rectilinear trajectory. The geometric relationship of the path in euclidean and curvilinear coordinates is shown at Fig. 2.

The coordinate origin of the base frame is defined as the closest point to the vehicle in the global path. The arc length of the base frame (s on the right image) is obtained as the distance of each point along the global plan (represented as a green line) to the coordinate origin. This distance is represented on the *x-axis* of the curvilinear system. On the *y-axis*, q represents the perpendicular lateral distance with respect to the path. The left side is represented by positive values and the right side by negative values.

For the computation of the transformation between the euclidean and the curvilinear coordinate system, path curvature κ is needed. This value is computed as follows [10], [5], [12]:

$$\kappa = \frac{S}{Q}\cdot\left(\kappa_b \cdot \frac{(1-q\cdot\kappa_b)\cdot(\partial^2 q/\partial s^2)+\kappa_b\cdot(\partial q/\partial s)^2}{Q^2}\right), \quad (2)$$

$$\text{where } \begin{cases} S = \text{sign}\,(1-q\cdot\kappa_b) \\ Q = \sqrt{\left(\frac{\partial q}{\partial s}\right)^2 + (1-q\cdot\kappa_b)^2} \end{cases} \quad (3)$$

There, κ_b is the curvature of the segment of the base trajectory used for the computation of the Frenét space. The candidate paths generation is performed in the curvilinear space, without considering the obstacles in the environment. These will be taken into account later, once the tentative trajectories are transformed to the euclidean space.

Maneuvering Paths Generation: The curvature of the generated paths is defined by the lateral offset q with respect to the base frame. First and second order derivatives of q are needed to compute κ (see equations 2 and 3), so a function dependent on the lateral offset is needed to compute a smooth lateral change.

q can be defined by a sequence of a cubic polynomial and a set of constants [10]:

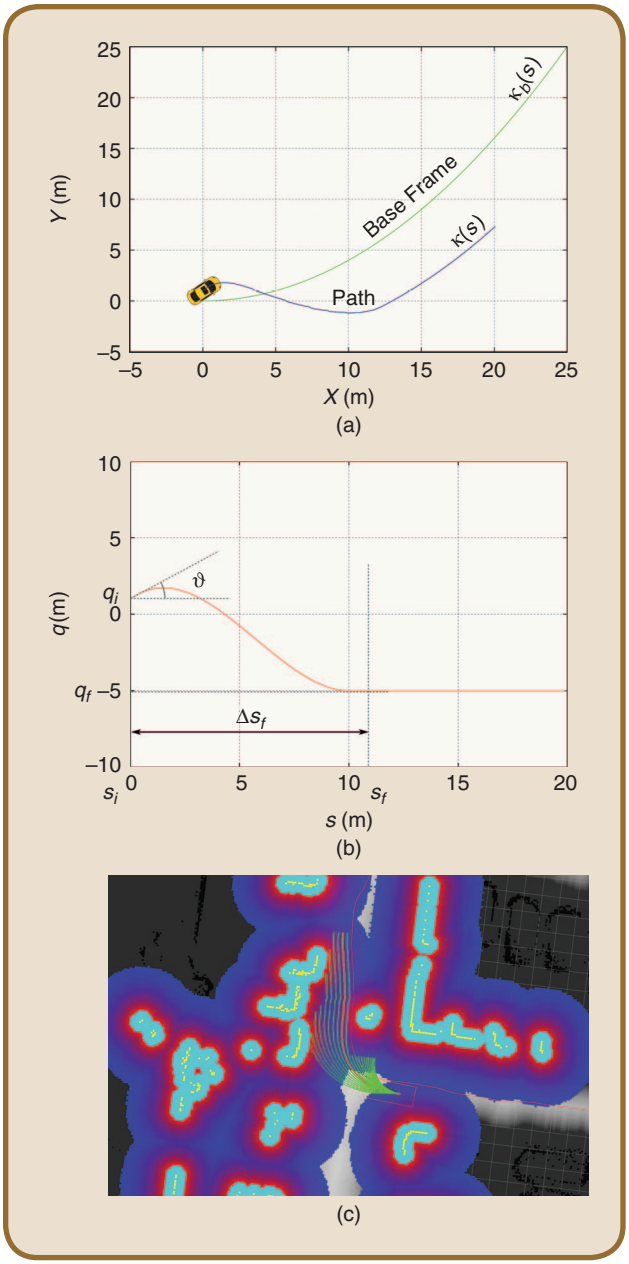


FIG 2 Conversion of a trajectory between the Cartesian and Frenét spaces, and paths truncation example.

$$q(s) = \begin{cases} a\cdot\Delta s^3 + b\cdot\Delta s^2 + c\cdot\Delta s + q_i & \text{if } s_i \le s < s_f \\ q_f & \text{if } s_f \le s \end{cases}$$
$$\frac{\partial q}{\partial s}(s) = \begin{cases} 3\cdot a\cdot\Delta s^2 + 2\cdot b\cdot\Delta s + c & \text{if } s_i \le s < s_f \\ 0 & \text{if } s_f \le s \end{cases}, \quad (4)$$
$$\frac{\partial^2 q}{\partial s^2}(s) = \begin{cases} 6\cdot a\cdot\Delta s + 2\cdot b & \text{if } s_i \le s < s_f \\ 0 & \text{if } s_f \le s \end{cases}$$

where $\Delta s = s - s_i$.

In Fig. 2b, the components involved in this process are depicted.

- The initial length s_i is zero, since the global planner is being pruned as the vehicle advances. Lateral offset q_i with respect to the global path's origin is also known.
- Angle θ defines the difference between the vehicle heading angle and the tangent angle of the base frame at the current position. If $\theta > 40°$, the vehicle enters in a recovery state, described in section III-B3. Once the vehicle is headed properly towards the path, this recovery state ends.
- s_f is a parameter that controls the longitudinal distance needed to reach offset q_f. This distance should be dependent on the speed. However, as the top speed of the prototype is not too high, Δs_f can be considered as the distance needed to go from q_i to the biggest q_f at top speed.
- The different q_f are computed separately for each path attending to the parameters defined by the user. In our implementation, the method receives as input the maximal width covered between the outer left and the outer right generated path $w_{\max}$, and the total number of paths to be generated (n_{paths}). So the value of q_f is computed as $w_{\max}/(n_{\text{paths}} - 1)$. s_f is also a free parameter provided directly by the user as the maximal longitudinal length that is desired to be covered by the paths.

Once the paths are generated in the curvilinear coordinate system, they are transformed to the euclidean space. In this new space, it is possible to evaluate their corresponding costs.

To define the maximum length of each candidate path, the cost of the cells corresponding to the points in the trajectory are inspected. If this cost is over threshold $\tau_{\text{circumscribed}}$, (which is the cost of cells at a distance equal to the radius of the minimum circumference that contains the outer limits of the vehicle footprint), the path is truncated at this point, as shown in Fig. 2c, where the generated paths are shown together with a colored costmap representation. In this figure, blue represents a low cost value, while the red color is used for the higher costs. Yellow and cyan correspond to lethal cells and inscribed cells, respectively.

2) Winner Path Selection

At each iteration, the winner path is selected through the use of a linear combination $J[i]$ of weighted cost functions, related to the following parameters: occlusion, length, distance to the global path, curvature and consistency of the path. $J[i]$ is evaluated as follows:

$$J[i] = \omega_o C_o[i] + \omega_l C_l[i] + \omega_d C_d[i] + \omega_\kappa C_\kappa[i] + \omega_c C_c[i] \quad (5)$$

Here, i is the path index, and C_o, C_l, C_d, C_κ and C_c are the costs of occlusion, length, distance to the global path, curvature and consistency, respectively. Their relative factors ω_k, $k \in \{o, l, d, \kappa, c\}$ are the associated weights that allow to adjust the influence of each of the costs to the final cost value.

The following costs will be computed for each candidate path independently in the euclidean space.

a) Occlusion

The occlusion cost is related to the safety of the path. This cost estimates the goodness of a path, such that the best paths are those which pass far enough from the obstacles. To do so, the footprint of the vehicle is simulated in each point of the path. The occlusion cost corresponds to the normalized maximum cost along the path:

$$C_o = \frac{\max\{c_i\}}{255}, \qquad i = 1 \ldots L \quad (6)$$

In this expression, L is the length of the current path being evaluated. $\max\{c_i\}$ is the maximum value of all the costs, associated to a point in the path.

b) Length

This cost is related to the length of the current path. The longer the path is, the lower its associated cost is. In general, a long path implies the existence of an area which is free of obstacles and can be safely traversed.

$$C_l = 1 - \frac{\sum_{i=1}^{L} \| p_i - p_{i-1} \|}{q_{f_{\max}} + s_f} \quad (7)$$

Here, p_i is a certain point inside the evaluated path. $q_{f_{\max}}$ is the maximum value that q_f can have for a certain path. Lengths are normalized to a value that a path will never reach.

c) Distance to the Global Path

This cost represents the lateral offset of the vehicle with respect to the global path. Tuning the associated weight of this cost will change the behavior of the vehicle when returning to the global path, after an occasional obstacle is avoided. It is computed as follows:

$$C_d = \frac{\sum_{i=1}^{L} \| p_i - \text{nearest}(p_i, g) \|}{L \cdot q_{f_{\max}}}, \quad (8)$$

where nearest(p, g) is the nearest point in the global path g to point p. This cost is normalized with respect to the maximum expected offset, $q_{f_{\max}}$.

d) Curvature

This cost allows to give priority to smoother paths. Let $p(x_i, y_i), i = 1 \cdots L$, be a point in the path. Then,

$$C_\kappa = \max\left\{ \frac{x_i' \cdot y_i'' - x_i'' \cdot y_i'}{(x_i' + y_i')^{3/2}} \right\}, \qquad i = 1 \ldots L \quad (9)$$

e) Consistency

This cost avoids continuous changes in winner paths between iterations. Once the vehicle starts a maneuver, it is

preferable to keep the same behavior during the following iterations. This is done through the following expression:

$$C_c = \frac{1}{s_2 - s_1} \int_{s_1}^{s_2} l_i \, ds \tag{10}$$

Lateral cost $l_i(s)$ is the distance between the current path and the previous winner path, at the same longitudinal position s; s_1 and s_2 are the first and last positions over s for which there are common points in both trajectories. At the beginning of the trip, the oscillation cost is 0, so it does not affect to the final choice of the path.

Once all costs are computed, the expression described in equation 5 is applied. In those paths for which it is impossible to advance, due to the presence of a nearby obstacle, the cost is forced to a negative value in order to indicate the rest of the system that this path is invalid.

The path with the lowest cost is selected (winner path W). If for some reason there is no valid path, the vehicle stops until the road is free of obstacles. If this situation does not change for a while, the recovery behavior process starts.

3) Recovery behavior

There are two scenarios in which the vehicle executes a recovery maneuver. The first one occurs when the vehicle is not correctly aligned with respect to the global plan, and the initial angle is too large to produce feasible local paths that comply with the curvature restrictions of the vehicle. The second case happens when the vehicle is correctly aligned but there are no feasible local paths to follow, due to the presence of a nearby obstacle.

In the first case, the recovery maneuver is intended to align the vehicle to the global path again. Same as for the local planner, the recovery behavior of the vehicle is composed of

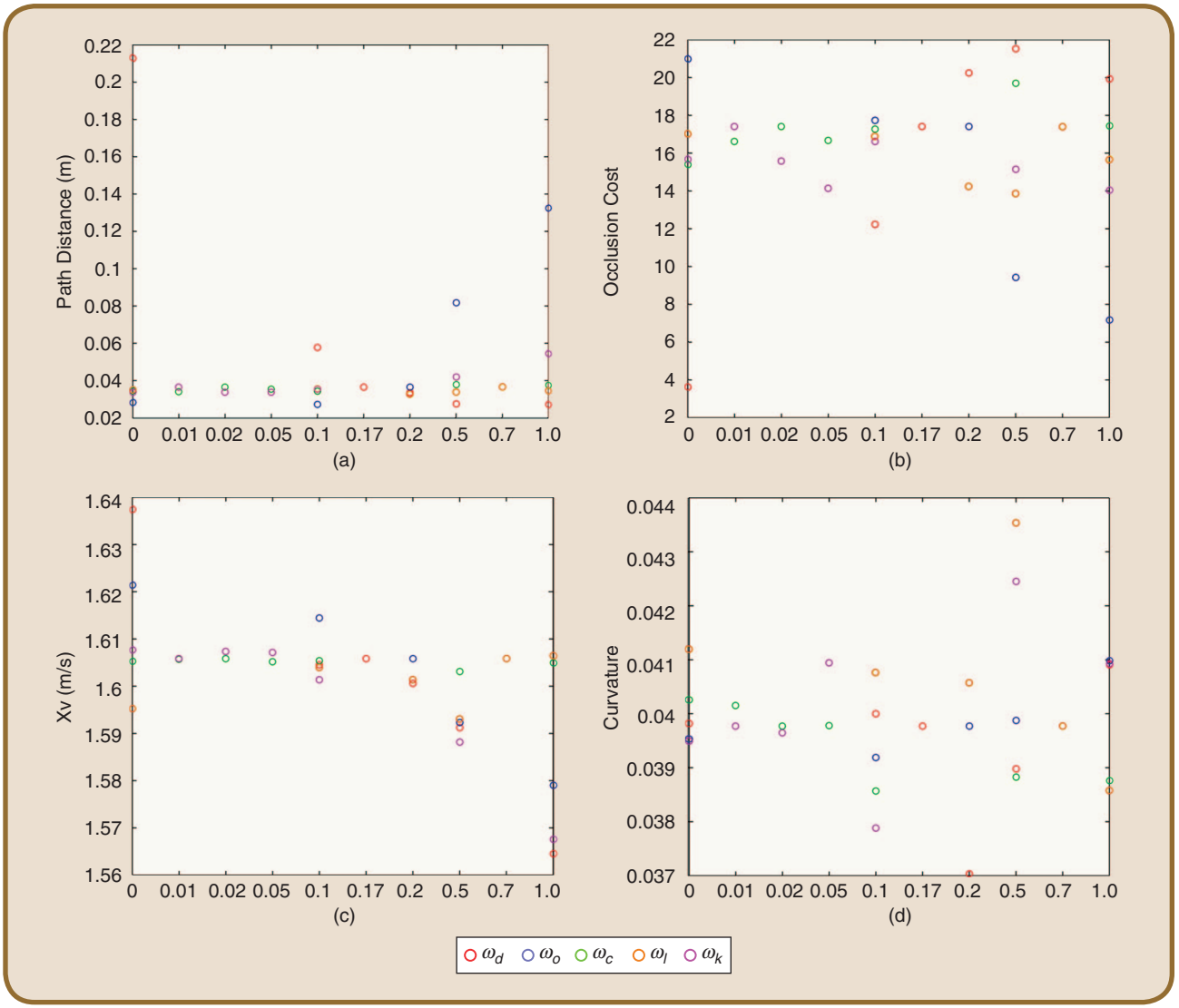


FIG 3 Results obtained from parameters.

two main phases. In the first phase, a set of feasible paths is generated. In the second phase, these paths are weighed in order to choose the best option. The vehicle will then try to follow the winner path. The recovery paths are chosen among four options: two forward paths and two backwards paths, setting the steering wheel to the maximum allowed angle at both left and right sides. The recovery maneuver is composed of a sequence of one or more of these paths.

In the second case, if possible, the vehicle just moves backwards for a short distance, in order to obtain enough space for the generation of feasible local paths.

IV. Results

In this section, the behavior of the local planner is described, using different cost weights as input.

A. Experimental Setup

In order to study the behavior of our method, several trajectories were followed, while recording a set of measured variables. These variables were *distance to path*, which measures the distance from the center of the vehicle to the closest point in the global path; *occlusion cost*, which measures the maximum cost of the cells below the vehicle footprint at each iteration; the *speed*, assuming that faster trajectories are preferred; and the *curvature* of the followed trajectory.

Since keeping the exact same conditions for all the tests is desirable, a simulator was used. In each test, the vehicle started at the exact same position and traveled towards the exact same goal. Obstacles were always in the same locations, and the only changing values were the input parameters under evaluation. The obtained results were then validated with some tests under real conditions, using the Verdino platform.

As seen in section III-B2, there are five different parameters that influence the overall cost, which will determine the chosen winner path. Each parameter has an associated weight. A base configuration of $\omega_d = 0.17, \omega_o = 0.2,\ \omega_c = 0.02, \omega_l = 0.7, \omega_k = 0.01$, which

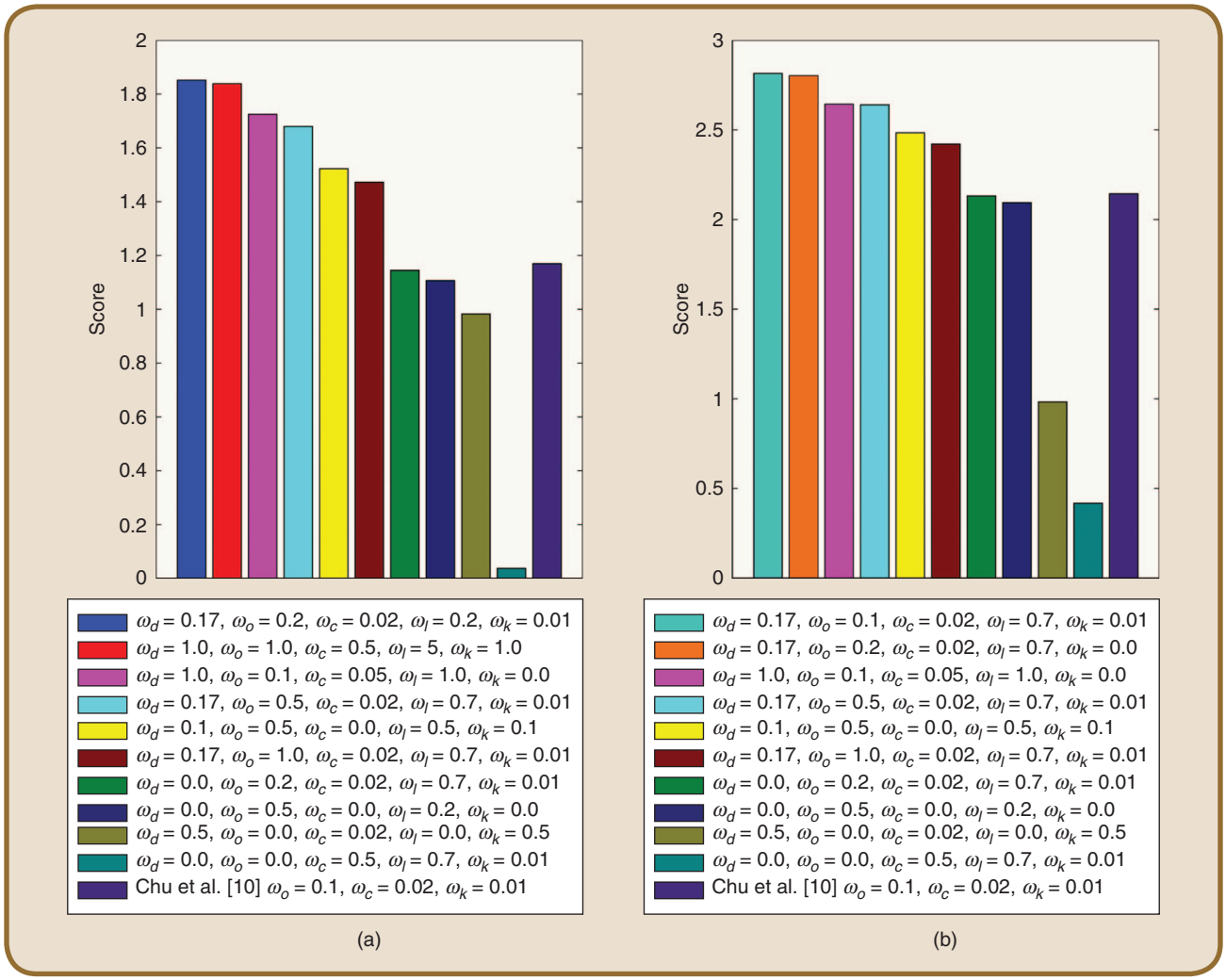


FIG 4 Results obtained from rankings.

provided good empirical results, has been used. Using the base configuration as a starting point, different weight configurations have been obtained by incrementally varying each weight.

Fig. 3 shows how the variation of each individual weight influences the measured variables. In order to establish the relative importance of each weight with respect to the base configuration, for each performed test one of them varies in the [0, 1] range, while the rest keep their default values. Fig. 4 shows a ranking with different cost weights configurations. In Fig. 4a, the evaluation has been done by taking into account the path distance and the occlusion cost. In Fig. 4b, the evaluation takes into account the occlusion cost and path distance again, but also the velocity and the curvature of the followed path. Verdino is intended to be used in pedestrian areas. If the speed and curvature costs are taken into account when computing the overall cost of the trajectories, paths with high speed and predominantly straight are more likely to be selected. This means that the vehicle will be more aggressive and less capable of maneuvering. In crowded environments, it is usually better to take a longer, slower path that skirts obstacles (people) by a large margin, than a fast, straight path that traverses near obstacles. For this reason, in order to reach a compromise between speed and maneuverability, the

> The recovery paths are chosen among four options: two forward paths and two backwards paths, setting the steering wheel to the maximum allowed angle at both left and right sides.

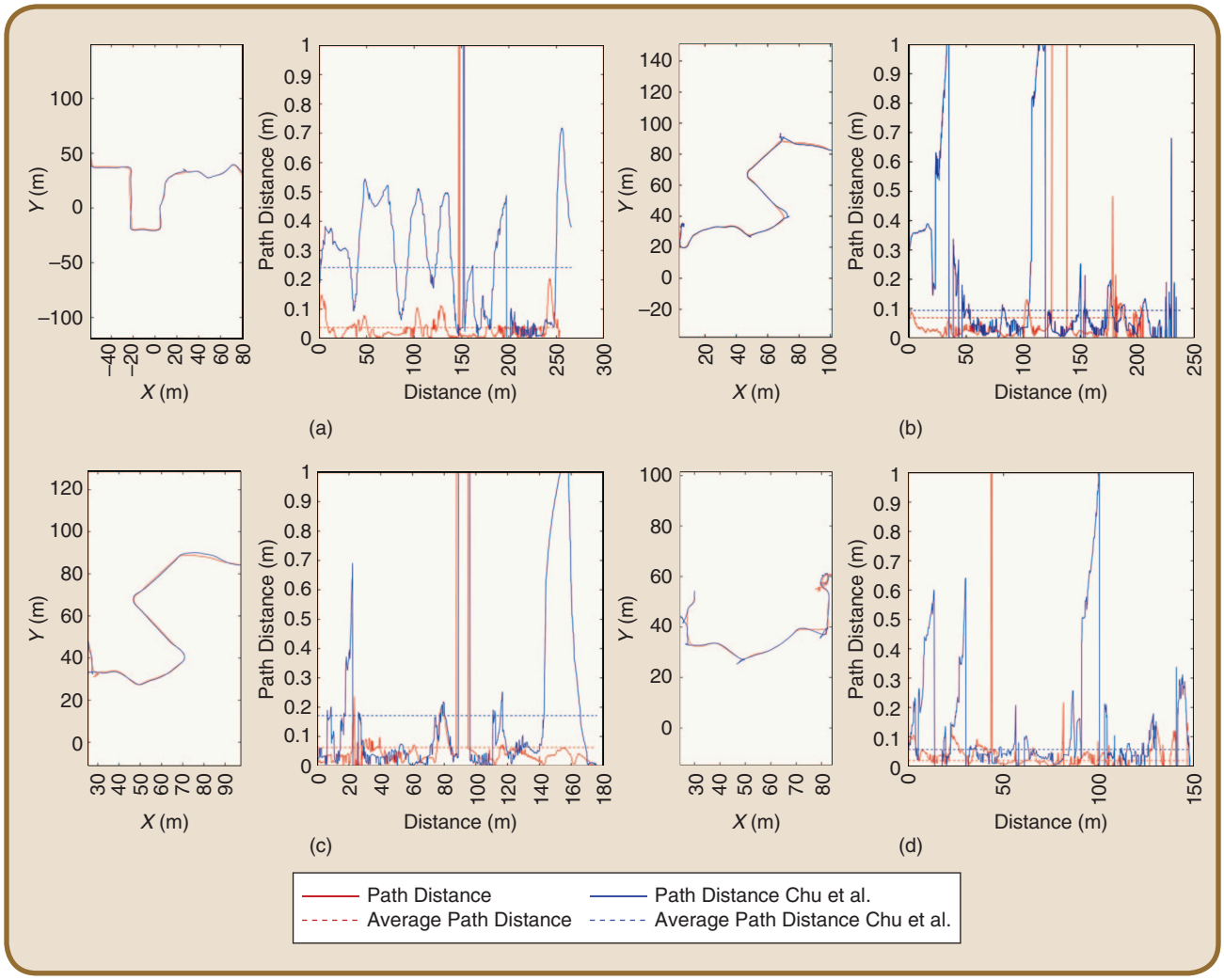


FIG 5 Traversed routes comparison of the proposed method (red) and the method proposed in Chu et al. [10] (blue).

The curvature of the trajectory is influenced by a combination of the different cost weights, and no particular cost influences it predominantly.

measured speed and the curvature are weighted by 0.5 to lessen their influence.

In order to aggregate variables with different units, all measured variables are normalized between 0 and 1 using the maximum and minimum values for each range. Also, prior to aggregating results, some variables are inverted. For example, the occlusion cost and the path distance are to be minimized, whereas speed and path length are to be maximized.

B. Cost weights determination

As expected, the measured variables are observed to be more greatly affected when varying their corresponding cost weight, whereas the influence of the other weights is not as evident. For instance, as Fig. 3a shows, varying the path distance cost causes the path distance variable to change proportionally. As Fig. 3b shows, the relationship between this cost weight and the occlusion cost is inversely proportional. This behavior may be produced due to the global plan traversing relatively high cost areas, mainly in closed curves. This occurs because the global planner does not take the vehicle dynamics into account when computing the global plan. When the cost weight of the path distance is high, the vehicle is forced to stick to the global path as much as possible and thus the cost can raise. If the occlusion cost weight is high, the vehicle's priority is to avoid obstacles as much as possible, so the distance to the global path grows.

Additionally, if the path length cost weight increases, the selection of long paths is favored. As the velocity command is computed as inversely proportional to the cost, in general, higher speeds are obtained when paths are longer (Fig. 3c). The navigation speed also depends on the distance to the obstacles, which is why the occlusion cost weight also influences this measurement. Moreover, if the curvature cost weight is high, the cost of the winner path in closed curves also increases. As speed is inversely proportional to the cost of the winner path, this causes the vehicle to slow down.

The curvature of the trajectory is influenced by a combination of the different cost weights, and no particular cost influences it predominantly (see Fig. 3d).

The rankings of Fig. 4a and Fig. 4b show which weight configurations produce the best results, based on different criteria. In Fig. 4a the ranking takes into account the occlusion cost and the path distance. In this sense, the configurations in the first places of the ranking favor mainly following the global path and avoiding obstacles. In the ranking shown in Fig. 4b the speed achieved by the vehicle and the curvature of the trajectory are also taken into account.

For the ranking in Fig. 4a, the best configuration was $\omega_d = 0.17, \omega_o = 0.2, \omega_c = 0.02, \omega_l = 0.2,$ and $\omega_k = 0.01$ (shown in blue color); and for the ranking in Fig. 4b, this configuration was $\omega_d = 0.17, \omega_o = 0.1, \omega_c = 0.02, \omega_l = 0.7,$ and $\omega_k = 0.01$ (shown in light blue color).

The method presented in [10] uses three weights: occlusion, curvature and consistency. For the sake of comparison, the path distance and path length costs weights have been set to zero in order to replicate the configuration used in [10]. This configuration is shown in violet color in Figs. 4a and 4b. As can be seen, there are several configurations with five costs that outperform the configuration with three costs proposed in [10]. As depicted in the charts, the use of the additional cost weights proposed in this work (ω_d and ω_l) produces better results in terms of vehicle behavior.

The charts of Fig. 5 compare the proposed method using the winner five-costs configuration of Fig. 4b (light blue color), to the method proposed in [10] that uses a three costs configuration, shown in violet color in Fig. 4b. On the left side of each figure, the traversed trajectory of the proposed method (red) and the method proposed in [10] (blue) are shown. On the right side of each figure, it is depicted the distance to the global path along the traversed trajectory of the presented method (red) and the method proposed on [10] (blue). Dashed lines show averaged path distance results.

As can be seen, the addition of the path distance costs makes the vehicle navigate closer to the global plan. Controlling the distance to the global plan is of capital importance in complex scenarios, like the ones Verdino is intended for. In these scenarios, mainly pedestrian areas, there are sharp turns and narrow navigable zones. Without a path distance cost, the vehicle may not follow the global plan properly. For example, when approaching a curve in which the vehicle passes from a wider to a narrower area, if the car predominantly selects paths with low occlusion costs, it will turn late. In sharp turns this can force the vehicle to initiate a recovery behavior to reorient itself, before it can continue following the global plan again. In the left side of the examples shown in Fig. 5, there are several situations where the method proposed in [10] has to

initiate a recovery behavior. In these situations, there is an abrupt change in the distance to the path, as shown in the right side of the examples. This occurs noticeably less frequently with the proposed method, which sticks properly to the global path.

Taking into account the obtained results, the winner configuration of Fig. 4b was chosen for the Verdino prototype. This method works at 10 *Hz*, on a *i7-3770K* processor, 16 Gb of *RAM DDR-3* memory, *SSD* storage and a NVIDIA GeForce GT 640. These times have been obtained under real conditions.

V. Conclusions

This paper presents a system which follows a global plan, by generating different tentative paths and choosing the most suitable one.

Different configurations have been tested and ranked, in order to select the parameters that will ultimately influence the vehicle behavior. These parameters are the length of the generated path, its distance to the global path, its proximity to obstacles, its curvature and its consistency. Using the obtained results, a winner configuration has been selected to be used in the real prototype Verdino.

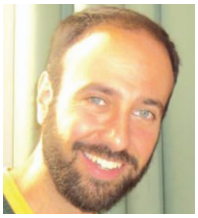
The results are compared to a similar method that uses three cost weights to select the local paths. As shown in the results section, the inclusion of two additional weights improve the navigation behavior of the prototype. This is the main contribution of the presented method with respect to the method presented in [10]. The inclusion of the path distance cost makes the vehicle follow the global plan more accurately than in the previous work presented in [10]. Additionally, the path length cost allows us to control the influence that the length of the winner path has in the speed of the vehicle. Another advantage is the use of a recovery behavior to overcome unforeseen situations, complementing the local planner and ensuring that the vehicle never gets stuck.

However, there is still room for improvement in the presented approach. One of the main drawbacks of the method is that if the angle between the vehicle and the path is too big, it is not possible to generate the paths, or they could not be followed by the vehicle due to physical restrictions. However, the use of the recovery maneuvers minimizes the impact of this limitation. Also, the obstacles are being considered as static, and no information about their previous motion is in use, which could lead to a more intelligent behavior of the vehicle. Although the iteration frequency of the method is high enough to reduce the effects of this lack of information, further research must be done in order to detect obstacles trajectories and to be able to include them in the costmap.

Acknowledgment

This work was supported by the project STIRPE DPI2013-46897-C2-1-R and the funds from the Agencia Canaria de Investigación, Innovación y Sociedad de la Información (ACIISI), cofinanced by FEDER funds (EU).

About the Authors



Dr. Rafael Arnay was born in 1982. He received the M.Sc. and Ph.D. degrees (with honors) in Computer Science from the University of La Laguna (Spain), in 2007 and 2014, respectively. He joined the Department of Computer Science and Systems, University of La Laguna in 2007, where he is currently a Postdoctoral Researcher. His current research interests include unsupervised learning, object segmentation/recognition, feature extraction and deep learning.

Dr. Néstor Morales is a researcher at Universidad de La Laguna. He received his Ph.D. in Computer Sciences on 2014, and his Computer Sciences Engineer degree on 2007. His PhD was related to different techniques for computer vision based obstacle detection for autonomous vehicles both using monocular and stereo vision, and the ways used for their avoidance. He has published several papers in several national and international conferences and journals. His research interests are computer vision, obstacle detection and tracking, and mobile robots planning.

Antonio Morell was born in 1981 in Santa Cruz de Tenerife, Spain. He received his B.Sc. degree in Computer Science (Hardware Engineering) in 2006 and his B.Sc. degree in Automation and Industrial Electronics Engineering in 2010 from the University of La Laguna (ULL), Spain. After receiving his M.Sc. degree in Research, Development and Innovation in Science and Engineering in 2011 also from the ULL, he worked as a researcher in Project SAGENIA, funded by the Spanish Ministry of Economy and Competitiveness until 2013. Since then, he is a Research Assistant supported with a grant from the Canarian Agency for Research, Innovation and Information Society (ACIISI), pursuing a Ph.D. in Engineering Physics. His research interests include sensor fusion, mobile robots, autonomous navigation, biped locomotion and robot modeling and control.

Javier Hernández-Aceituno was born in 1986 in Santa Cruz de Tenerife, Spain. He graduated in Computer

Engineering in 2010 at the Universidad de La Laguna (ULL), Spain. He is currently conducting his research as a Ph.D. student in ULL. His current research includes mapping, navigation, and obstacle detection algorithms for autonomous vehicles.

Daniel Perea Ström was born in Stockholm, Sweden. He received the B.Eng. degree in computer science and systems engineering in 2007, and the M. Eng. degree in computer science engineering from the University of La Laguna, in 2008. Since 2007 he is with the Robotics Group of the University of La Laguna, conducting research in the field of mobile robotics. Since 2009 he is developer of the self driving car Verdino in charge of the localization and mapping systems. His research interests are in the areas of robot navigation, simultaneous localization and mapping, exploration, and machine learning approaches.

Dr. Jonay Toledo is an Assistant Professor at University of La Laguna (ULL). He received his Master in Computer Science in 2001, Master in Electronics in 2002 and Ph.D. in Automatic Control in 2008. His current research interests include mobile robots, autonomous vehicles, automatic control and embedded systems. He is part of the development team of Verdino, an autonomous electric car designed to drive autonomously in pedestrian areas. He has published several conference and journal papers in robotics, automatic control, artificial intelligent and autonomous cars.

Dr. Alberto Hamilton was born in 1968. He received the M. Sc. in 1991 in Physics from University Complutense of Madrid and Ph.D. in Computer Science in 1997 from University of La Laguna (Spain). He joined the Department of Computer Science and Systems, University of La Laguna in 1991, where he is currently senior lecturer. His current research interests include mobile robotics, intelligent agents and pervasive computing.

Dr. Javier Sánchez-Medina earned his Engineering Master Degree at the Telecommunications Faculty on 2002, and his PhD at the Computer Science Department on 2008. He is interested in the application of Evolutionary and

Parallel Computing techniques to Intelligent Transportation Systems. He has more than 20 international conference and 10 international journal papers. He is also very active as a volunteer of the IEEE ITS Society, where he has been serving in a number of different positions. Currently he is EiC of the ITS Podcast, the ITS Newsletter Vice-president of the IEEE ITSS's Spanish chapter and General Chair for IEEE ITSC2015.

Dr. Leopoldo Acosta is a Full Professor at the Universidad de La Laguna, Spain. He has been involved in several competitive nationally-funded research projects related to Artificial Intelligence and Robotics, four of them as project leader. He has directed six Ph.D.

References

[1] L. Acosta, J. Toledo, R. Arnay, J. Espelosín, N. Morales, D. Perea, and L. Moreno, "Verdino, prototipo eléctrico de vehículo autoguiado," in *XXXIII Jornadas de Automática*, Sept. 2012, pp. 729–736.

[2] D. Perea, J. Hernandez-Aceituno, A. Morell, J. Toledo, A. Hamilton, and L. Acosta, "MCL with sensor fusion based on a weighting mechanism versus a particle generation approach," in *Proc. 16th Int. IEEE Conf. Intelligent Transportation Systems (ITSC 2013)*, Oct. 2013, pp. 166–171.

[3] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Stanley: The robot that won the DARPA Grand Challenge," *J. Field Robot.*, vol. 23, no. 9, pp. 661–692, Sep. 2006.

[4] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, and S. Thrun, "Junior: The Stanford entry in the Urban Challenge," *J. Field Robot.*, vol. 25, no. 9, pp. 569–597, Sep. 2008.

[5] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a Frenét Frame," in *Proc. 2010 IEEE Int. Conf. Robotics and Automation*, May 2010, pp. 987–993.

[6] D. Ferguson, T. M. Howard, and M. Likhachev, "Motion planning in urban environments," *J Field Robot.*, vol. 25, no. 11-12, pp. 939–960, Nov. 2008.

[7] M. J. Van Nieuwstadt and R. M. Murray, "Real-time trajectory generation for differentially flat systems," *Int. J. Robust Nonlinear Contr.*, vol. 8, no. 11, pp. 995–1020, Sep. 1998.

[8] S. M. LaValle, "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, May 2001.

[9] Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, J. How, and G. Fiore, "Real-time motion planning with applications to autonomous urban driving," *IEEE Trans. Contr. Syst. Technol.*, vol. 17, no. 5, pp. 1105–1118, Sep. 2009.

[10] K. Chu, M. Lee, and M. Sunwoo, "Local path planning for off-road autonomous driving with avoidance of static obstacles," *IEEE Trans. Intell. Transport. Syst.*, vol. 13, no. 4, pp. 1599–1616, Dec. 2012.

[11] D. V. Lu, D. Hershberger, and W. D. Smart, "Layered costmaps for context-sensitive navigation," in *Proc. Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ Int. Conf.*, 2014, pp. 709–715.

[12] T. Barfoot and C. Clark, "Motion planning for formations of mobile robots," *Robot. Autonomous Syst.*, vol. 46, no. 2, pp. 65–78, Feb. 2004.

ITS