An RRT-based Navigation Approach for Mobile Robots and Automated Vehicles

Luis Garrote, Cristiano Premebida, Marco Silva and Urbano Nunes
Institute of Systems and Robotics
Dep. of Electrical and Computer Engineering - University of Coimbra
Coimbra, Portugal
{garrote, cpremebida, msilva, urbano}@isr.uc.pt

***Abstract*—Advances in autonomous navigation, safety, and natural-landmark based localization, are among the key objectives in the development of the next generation of autonomous vehicles, to be deployed in manufacturing and semi-structured environments. In this paper, autonomous navigation and collision detection will be focused, where it is proposed a novel navigation approach that incorporates a RRT-based dynamic path planning and a path-following controller. Safety issues are taken into account in the form of a laser-based object detection and tracking. Experimental results obtained in a virtual environment provide evidence that our proposed navigation method is promising for real-world applications.**

I. Introduction

Recent advances in automated guided vehicles (AGVs), or laser guided vehicles (LGVs), are towards using AGVs beyond industrial environments, such as: hospitals, airports, offices, and intelligent transportation systems. AGVs and mobile robots share, in general, common issues regarding localization [1], trajectory planning [2], path-following, obstacle avoidance, local navigation [3], communication, sensor fusion, among others. However, some current applications impose new requirements for real-world cases, namely, dynamic path planning, navigation in cluttered environments, and human safety.

Many studies have targeted different aspects of autonomous vehicles with non-holonomic constraints, such as: kinematics, dynamics, localization, navigation (planning and path-following), controller design, and obstacle avoidance. In this paper the problem of safe navigation is under focus. Path-following and path planning under safety constraints (collision avoidance) are particularly addressed.

Path-following has the goal of minimizing a given lateral error distance between the vehicle and the defined trajectory (given *a-priori* or from a path planning algorithm). In general terms, this minimization is unbounded w.r.t. a time interval, while maintaining the pose error bounded. The path-following problem has been well studied in the mobile robotics community, and many solutions have been proposed and applied in a wide range of cases. Geometric solutions, including the control law used in the Stanley autonomous vehicle [4] and Pure Pursuit [5], were proven to be reliable and suitable to real world situations. In [2], sliding mode control was applied to differential and car-like robots, providing robust and stable control laws to systems under uncertainties and external disturbances. In [6] and [7], Fuzzy logic controllers were proposed and deployed successfully in autonomous vehicles.

Look Up this

Strictly related to path-following, path planning is a mature scientific area with proposed solutions ranging from incremental search (*e.g.*, A*, D* light, IDA*) to probabilistic algorithms (RRT, PRM). The later, probabilistic algorithms, have proven to be an efficient solution for path planning problems with kinodynamic constraints in non-convex higher dimensional spaces. In particular, Rapidly Exploring Random Trees (RRTs) [8] are among the most popular probabilistic approaches. RRT works by iteratively creating an exploration tree, starting by sampling nodes in unexplored regions of the search space, then finding suitable connections among nodes, and finally adding new nodes to the exploration tree. The process stops when a path is found or when a maximum number of iterations is reached. Building on the initial RRT framework, many methods have been proposed [9], [10] and applied successfully in real-world scenarios [11], [12].

The development of reliable systems for detection and tracking of moving objects (DATMO) is a key research goal in robotics and computer vision as it would greatly facilitate practical and real-world deployment of automatic safety systems for robots, AGVs, intelligent vehicles, and so on. Human detection, crucial in safety systems, saw much progress over the last decade evidenced by various works including [13], [14], [15]. Nevertheless, there is much room for improvement when algorithms are tested on realistic scenes, specially when solutions should be in compliance with safety requirements.

This work brings contributions in the navigation research field, where a novel approach incorporating a RRT-based planning and a path-following controller are presented. A laser scanner based DATMO system is addressed in section II. Section III-A details our RRT-based local path planning approach, including collision detection and obstacle avoidance. In section III-B our proposed path-following method is introduced. Discussion and experimental results using an in-house C/C++ based simulator are reported in section IV, before the paper concludes in section V.

II. DATMO

Assuming a single 2D laser scanner is mounted in the front part of an instrumented vehicle, this section deals with a

DATMO system which is composed of the following principal stages: (1) segmentation, (2) modeling, and (3) tracking.

1) Segmentation for obstacle detection: In general terms, segmentation is here defined as the process of separating foreground objects from the background in the laser scanner measurement space. The key step is to detect a breakpoint, characterized by a discontinuity between two consecutive laser points, which represents, possibly, an object boundary. Segmentation is a decisive processing phase since failures during this stage will strongly affect all the subsequent stages, with a not easy retrofit correction solution. A tradeoff exists in the segmentation process, which consists in deciding between merging or splitting the spatial distributed laser-points. In describing a segmentation process based on 2D range data, a given segment S_j is defined by the set of laser-points, sharing similar spatial properties that respect a given clustering condition. Therefore, S_j is the set of points conditioned on the parameters of a given (chosen) segmentation method.

Among several methods that can be used for 2D laser data segmentation, see [16] for a short review, here we briefly describe the segmentation method introduced in [17] which uses a stochastic filter. This method, called KF-based Breakpoint Detector (KFBD), uses the Kalman filter (KF) in conjunction with a statistical test, assuming a Chi-square validation region, to detect breakpoints. The stochastic model used to describe the spatial-dynamic evolution of the range measurements as well as the transition matrices are described in [17]. Basically, a breakpoint is detected if the *normalized innovation squared* exceeds a threshold Thr_χ according to a χ^2_1 distribution table. The model used in the KFBD method is not restrictive w.r.t. shape thus, it assumes a constant rate of change between the range-distance and the angle (also termed constant speed model).

2) Object modeling: Given the set of segments $\{S\}$: $S_j \in S$ obtained by the segmentation stage, at this point the purpose is to represent each segment S_j by a compact and proper geometric primitive. Among the possibilities (*e.g.*, line-segments, circle, rectangle, ellipse, quadrics), we chose circle as the primary primitive to represent a segment. Thus, after circle extraction using the method presented in [18], a segment S_j is represented by three parameters (x_c, y_c, rd): center and radius. Although circle is a suitable representation for many categories of objects (*e.g.*, poles, pillars, humans, small machines), in some cases rectangles can be used instead circles, specially when vehicle-like obstacles are detected. When time is under consideration, which is particularly important in the tracking stage, a segment will be explicitly represented by $S_j(k)$, otherwise the time-index (k) will be omitted.

3) Object tracking: Under linear discrete-time varying stochastic assumptions, objects dynamics are assumed to evolve according to a second-order kinematic model, driven by white noise. The position of a detected object/obstacle, assuming that it is defined by the circle center (x_c, y_c) extracted from a segment $S_j(k)$, is considered to evolve in time constrained to piecewise constant white noise acceleration [19] (constrained to maximum values of acceleration). This model can be understood, in more general terms, in the sense that the corresponding second-order derivative of the position is actually not zero (as in a theoretically noiseless model), but a zero-mean random process entering into the system in the form of random input *noise*.

The model assumptions, considering a sample-time interval h, are that the object keeps a constant acceleration during h: the noise processes are uncorrelated from period to period (piecewise indication). Using the KF and a global nearest-neighbor data association strategy, our tracking approach is posed in terms of a decoupled solution, that is, the motion along each coordinate is assumed *decoupled* (independent) from the other coordinates. More specifically, each coordinate (x, y) of the 2D-Cartesian space is governed by its own equation, with noises entering into x and y coordinates assumed to be mutually independent with possibly different variances.

III. Autonomous Navigation

In this section we introduce our autonomous navigation approach, which encompass dynamic path planning (based on the RRT* [9] algorithm), collision detection, and path-following. The functional block-diagram of the navigation system is shown in Fig. 1, having as inputs the set of tracked obstacles, the desired path and the vehicle pose (2D position and orientation).

When dealing with scenarios such as hospitals, airports, offices or structured industrial environments, and assuming a prior planned route/mission to follow, a number of non-systematic disturbances can arise. Static obstacles, people and objects (*e.g.*, vehicles) in motion require different behaviors to be taken into account by the navigation system. Using information from the DATMO in the form of tracked obstacles (see sec. II), knowing the vehicle pose and the desired path to follow, the navigation system is basically in charge of local dynamic path planning and path-following.

In situations of obstructed or infeasible paths, safety measures have to be ensured. In this regard, we defined a set of rules that constitute the baseline of our safety system for semi-structured environments:

- Minimum safe distance: the autonomous vehicle must keep a minimum distance from the infrastructure and obstacles in the pathway.
- Bounded lateral error: a small error while following a path must be guaranteed (< 0.5 m).
- Obstacle avoidance: avoidance maneuvers can only be taken if there is no risk of collision.

The rules listed above are continuously checked and, if an unsafe condition exists, the current local path is re-planned according to the dynamic path planning approach represented in Fig. 2. If a valid solution is found, the new (re-planed) path is sent to the path-following controller, otherwise the vehicle is safely immobilized.

A. Dynamic Path Planning

The dynamic path planning method proposed here is based on a modified version (summarized in Fig. 3) of the original

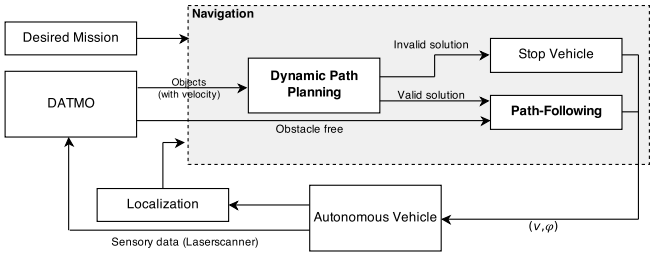


Fig. 1. Functional block-diagram of the vehicle's navigation system with the dynamic path planning and path-following blocks highlighted in bold.

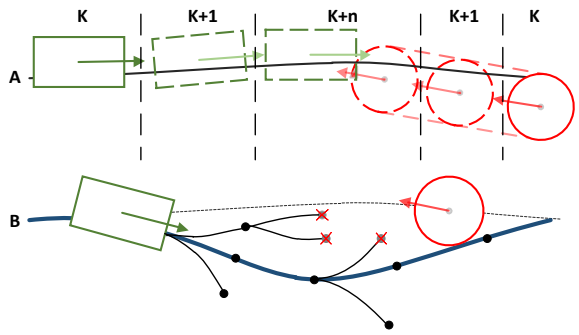


Fig. 2. The automated vehicle is represented by the green-box, while a detected object is represented by the red-circle. Estimations ahead are shown in dashed line. In the case of a predicted collision, a new valid path is created.

RRT* presented in [9]. The RRT* is a variant of the RRT algorithm that has an almost sure convergence to an optimal solution (asymptotic optimality property). As in the original RRT* algorithm, each node of the RRT contains the vehicle pose and the last control command. Nevertheless, we introduced two additional parameters: a time frame element and a penalization term. The time frame element is added to each RRT node for collision check (used in routine **CollisionFree** in Figs. 3 and 6) where a time estimate is maintained from the root node (beginning of RRT tree); this element is updated using the routine **UpdateTimeFrame** using the vehicle speed estimate (or speed profile) and the traveled distance between a given node and its parent node.

The penalization term, updated by routine **PenalizeNode**, contains the number of invalid expansions already performed and is used to assess if a node can be further explored. If the term exceeds a given threshold (in our experiments, a threshold of 5 failed explorations was defined) the node is considered to be unexplorable and assumed to be in a intersection zone. The intersection zone is here defined by all the nodes marked as unexplorable.

Furthermore, our modified sampling routine (**SampleFree**) uses the penalty term to bias the search towards the free space (and goal) and away from the intersection zone: pseudocode is shown in Fig. 4.

An important task in dynamic path planning is the capability to perform collision detection (with static and dynamic obstacles) and guide the search to a valid and collision-free path.

Input - Initial configuration : x_{init}
Output - RRT graph : G
G.init(x_{init})
for k=1 to K **do**
  $x_{rand} \leftarrow$ **SampleFree**(G)
  $x_{nearest} \leftarrow$ Nearest(x_{rand}, $G = (V, E)$)
  $x_{new} \leftarrow$ Steer($x_{nearest}$, x_{rand})
  UpdateTimeFrame($x_{nearest}$,x_{new})
  if CollisionFree($x_{nearest}$,x_{new}) **then**
    $radius \leftarrow \min\{\gamma_{RRT^*}(\log(card(V))/card(V))^{1/d}, \eta\}$
    $x_{near} \leftarrow$ Near($G = (V, E)$,x_{new},radius)
    $V \leftarrow V \bigcup \{x_{new}\}$
    $x_{min} \leftarrow x_{nearest}$
    c_{min} = Cost($x_{nearest}$)+Cost($x_{nearest}$,x_{new})
    for all $x \in x_{near}$ **do**
      if CollisionFree(x,x_{new}) $\bigwedge$ Cost(x) + Cost(x,x_{new}) < c_{min} **then**
        $x_{min} \leftarrow x$
        $c_{min} \leftarrow$ Cost(x)+Cost(x,x_{new})
        UpdateTimeFrame(x_{min},x_{new})
        $E = E \bigcup \{(x_{min},x_{new})\}$
      end if
    end for
    for all $x \in x_{near}$ **do**
      if (**CollisionFree**(x,x_{new}) $\bigwedge$ Cost(x_{new}) + Cost(x,x_{new}) < Cost(x) **then**
        $x_{newParent} \leftarrow$ Parent(x)
        UpdateTimeFrame(x_{new},x)
        $E \leftarrow (E \setminus \{x_{newParent},x\}) \bigcup \{(x_{new},x)\}$
      end if
    end for
  else
    PenalizeNode($x_{nearest}$)
  end if
end for

Fig. 3. RRT* algorithm adapted from [9] - in this paper the routines **PenalizeNode** and **UpdateTimeFrame** were introduced while the routines **CollisionFree** and **SampleFree** from RRT* were modified.

Input - RRT tree : G
Output - RRT node x_{rand}
$x_{rand} \leftarrow \emptyset$
while Penalization(Nearest(x_{rand})) > $MaxPenalization$ **do**
  $x_{rand} \leftarrow$ **RandomNode**($G.Goal$)
end while

Fig. 4. **SampleFree** altered routine uses the nearest node (within a search threshold) to assess if a new node is a RRT node candidate.

Focusing on collision problems like the one described in [20], in this paper the proposed approach for collision check (summarized by routine **CollisionFree** in Fig. 6), was implemented in such a way that for each node expansion on the RRT algorithm the dynamic behavior of obstacles is taken into account. Moreover, each RRT-node contains motion information (last control command and time frame), meaning that from a node to another, the position of the vehicle can be predicted with a bearable uncertainty margin. In particular, we assumed that a given detected object/obstacle is geometrically represented by a circle and by its estimated velocity (as previously detailed in sect. II-1). On the other hand, as illustrated in Fig. 5, the vehicle is defined by a 2D rectangle where the boundaries (vehicle width and length) are extended by a factor that depends on the safe emergency stoppage Δ_s criterion (see

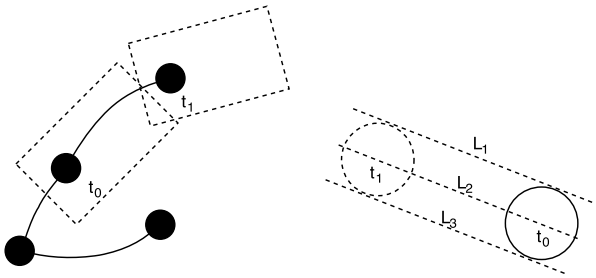


Fig. 5. **CollisionFree** - When testing if a new edge of the RRT is collision free, projections from the parent-node (at t_0) to the expanded node ahead (at t_1) are checked.

1: **Input** - Parent Node : $x_{nearest}$
2: New Node : x_{new}
3: **Output** - Collision State : $state$
4: $state \leftarrow false$
5: $box \leftarrow$ ComputeVehicleBoundingBox($x_{nearest}$,x_{new}) [eq. 4]
6: **for all** $Detected\ Obstacles\ i$ (DATMO) **do**
7: **if** i in $motion$ **then**
8: L_1,L_2,$L_3 \leftarrow$ Projection(i,Time($x_{nearest}$),Time(x_{new})) [Fig. 5]
state $\leftarrow$ HasIntersection(box,L_1,L_2,L_3)
9: **else**
10: $state \leftarrow$ HasIntersection($x_{nearest}$,x_{new},i)
11: **end if**
12: **if** $state\ is\ true$ **then**
13: return
14: **end if**
15: **end for**

Fig. 6. Our **CollisionFree** routine checks if two nodes ($x_{nearest}$ at t_0 and x_{new} at t_1 in Fig. 5) collide with any of the detected obstacles.

eq. 4). Under ideal conditions, the autonomous vehicle motion is described by:

$$v_{k+1} = v_k + a_k T \tag{1}$$

where v is the vehicle velocity, a is the vehicle acceleration, T is the time interval ($T = T_{k+1} - T_k$), with (1) being constrained by an upper velocity (v_{max}): $\|v_k + a_k T\| \leq v_{max}$. The vehicle incremental displacement is then given by:

$$\Delta = v_k T + \frac{1}{2} a_k T^2 \tag{2}$$

If a collision is imminent or an emergency occurs, the vehicle is forced to stop. Under deterministic assumptions, the vehicle stops completely when $v_{k+1} = 0$, thus:

$$0 = v_k + a_k T \tag{3}$$

Replacing (3) in (2), the safety distance is given by

$$\Delta_s = \frac{v_k^2}{2a_k} \tag{4}$$

Incorporating a safety distance into the planner, as presented above, makes the collision detection performance more feasible. From a parent node (at t_0) to a new node (at t_1), as exemplified in Fig. 5, the collision detection algorithm checks for potential intersections between the vehicle bounding box and a moving obstacle. Considering an approaching obstacle (dashed circles), and the vehicle predicted motion (dashed rectangles at t_0 and t_1): the three line segments (L_1, L_2, L_3) and the predicted vehicle boundaries (generated during the **Projection** and **ComputeVehicleBoundingBox** routines (Fig. 6)) are then tested for intersection. This procedure is used at each RRT iteration, and without a high computational burden the dynamic path planning can produce feasible solutions in dynamic scenarios.

B. Path-following Using Motion Primitives

This section presents our novel framework for path-following using sampled motion primitives based on the vehicle kinematic model. In this paper, we consider the kinematic model of a car-like vehicle as detailed in [21], with estimated position and orientation given by:

$$X_{k+1}(u) = \begin{cases} x_{k+1} = x_k + \cos\theta_k v \\ y_{k+1} = y_k + \sin\theta_k v \\ \theta_{k+1} = \theta_k + \frac{\tan\varphi_k}{L} v \end{cases} \tag{5}$$

where (x, y) designates the 2D position of vehicle's rear axle center, θ is the orientation angle w.r.t. the x axis, L is the distance between axles, φ gives the steering angle and v is the vehicle speed. This model is subject to constraints $\|\varphi\| \leq \varphi_{max}$ and $v_{min} \leq v \leq v_{max}$.

The trajectory of an autonomous vehicle can be described in terms of a set of motion primitives. These primitive motions (shown in Fig. 7) result from the control commands applied to the vehicle and, in the sense of path-following, the best control command is the one that minimizes the error between the vehicle estimated position and the local desired path.

The path-following approach proposed in this work is described in terms of two sets, $M(u) = \{X_1, X_2, ...X_n\}$ and $P = \{\mathbf{p}_1, \mathbf{p}_2, ...\mathbf{p}_g\}$ with respective set sizes n and g. $M(u)$ is a primitive motion, with length l_n (in meters), that corresponds to the dynamic evolution of the vehicle (eq. 5) according to the control input ($u = \{v, \varphi\}$). P is the desired path, within a preview window, corresponding to the set of points from the control point (the center of the rear axle) projected on the road with length l_g (in meters).

Assuming the set U comprises all control input configurations, and the set U' is a sampled subset of U containing all plausible control input configurations, then our path-following control law is given by the control input $u_c \in U'$ that minimizes the error between all projected motion primitives and the desired path:

$$u_c = K_1 \min_{u \epsilon U'} \{h(M(u), P)\} \tag{6}$$

where $h(M(u), P)$ is a measure that represents the similarity between two sets (*e.g.*, Hausdorff distance [22]) and K_1 a proportional gain. The resultant is the vehicle desired command u_c=(v_c, φ_c). Assuming the autonomous vehicle is moving locally at constant speed and that the allowed speeds are given as function of the path curvatures, the sampling space can be restricted to a limited number of acceptable steering

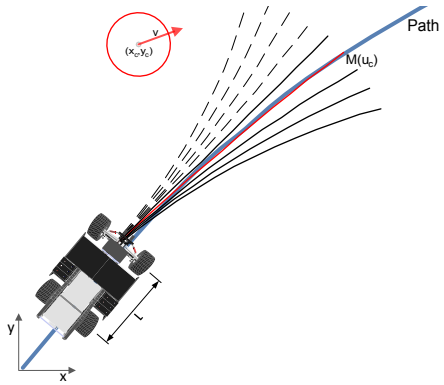


Fig. 7. Motion primitives (black lines) obtained from the kinematic model of a car-like vehicle (Ackerman steering). Desired path is represented by the solid blue line, while the optimal motion primitive is shown in red.

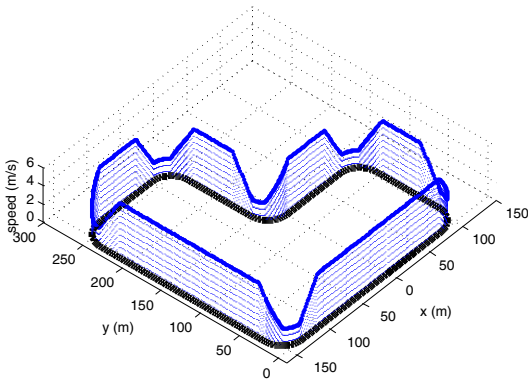


Fig. 8. A path-following simulation scenario with the defined path given in black, and the speed profile shown in blue.

angles φ, where this number decreases proportionally as the vehicle speed increases. Finally, considering the outputs from an obstacle detection system, motion primitives that lead to a risk of collision are discarded: dashed-lines in Fig. 7.

IV. Experiments

The evaluation of the path-following method presented in this paper is based on the vehicle model detailed in [6]. The autonomous vehicle model was simulated with a maximum speed of 9 m/s (32.4 Km/h). For performance comparison purposes, the proposed method is compared with three well stablished methods:

Method 1: having the control law given by [4]:

$$\varphi_c = e_\theta + \arctan(K_1 \frac{e_l}{v}) \tag{7}$$

where e_θ is the angular error, e_l is the lateral error, K_1 is a gain parameter and v is the vehicle speed.

Method 2: with control law of the form [5]:

$$\varphi_c = \arctan(K_1 \frac{2Le_l}{v^2}) \tag{8}$$

where e_l is the lateral error, L is the distance between axles, K_1 is a gain parameter and v is the vehicle speed.

Method 3: this control law is expressed by [6]:

$$\varphi_c = Fuzzy\,System(e_l, e_\theta) \tag{9}$$

where e_θ is the angular error and e_l is the lateral error. All angular and lateral errors are computed taking into account a lookahead distance (La). The lookahead distance is the distance from the control point (located in the center of rear axle) to a virtual control point positioned in front of the vehicle. The lookahead approach and the fuzzy path-following framework (summarized by the function $Fuzzy\,System$) are explained in [6].

To provide a suitable evaluation, four performance measures are used to compare the path-following controllers: maximum lateral error (MLE), control effort (CE), mean square error (MSE) and smoothness variation (SV). MLE is the maximum measured error between the rear end of the vehicle and the desired path. MSE is a measure of controller performance in terms of lateral error, CE is defined by the average of all absolute control commands demanded by the controller. Finally, SV represents the variation of the control law, where small variations means stable transitions. An initial experiment was conducted in a simulated *Matlab*®-based scenario (given in Fig. 8) with tight curves and different speed profiles. Considering this path, the parameters of the evaluated methods were empirically adjusted:

- Method 1 : $K_1 = 1.6$; $L_a = 3.0\,m$
- Method 2 : $K_1 = 0.02$; $L_a = 2.16\,m$
- Proposed : $K_1 = 0.95$; $l_n = l_g = 1.5v$

TABLE I
Path-following results considering the speed profile and the path given in Fig. 8.

	MLE	MSE	CE	SV
Method 1 [4]	0.0940	0.0022	0.0338	5.5e-04
Method 2 [5]	0.2179	0.0016	0.0366	0.0015
Method 3 [6]	0.7655	0.0825	0.1051	0.0176
Proposed	0.0856	1.6e-04	0.0351	9.7e-04

Considering the path shown in Fig. 8, the results obtained by the path-following methods are summarized in Table I, where all methods produced smooth trajectories and adapted well to the defined speed profile. The Method 1 achieved, in general, a performance as good as the Method 2, but the later had a higher MLE. On the other hand, Method 3 showed the highest lateral error because of the controller project (fuzzy variables and rules were designed for low speeds in the range $0.5 - 2m/s$ [6]). Finally our proposed controller maintained stable trajectories with small errors in terms of MLE and MSE.

In order to evaluate the behavior of the navigation approach, experiments were carried out in a in-house 3D C/C++ simulation software using the road-like scenario shown in Fig. 9. A desired path was defined and a set of runs were executed. Using our RRT-based algorithm (Fig. 3), the trajectories performed by the proposed path-follower, with obstacle free, are given in Fig. 9. Lastly, further experiments were performed

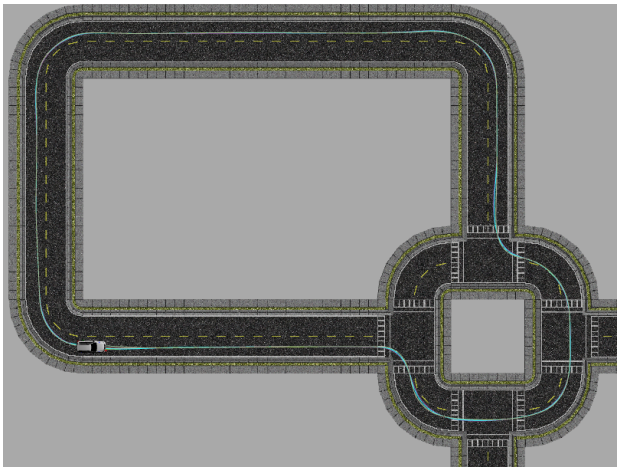
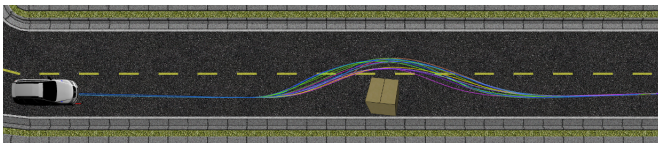
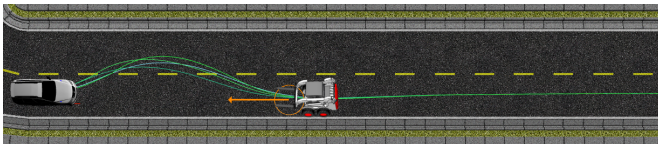


Fig. 9. An overview of simulation experiments conducted using our 3D simulation software for the evaluation of the proposed navigation solution.



(a) Static obstacle



(b) Moving obstacle

Fig. 10. Navigation approach behavior, using a RRT-based algorithm and a novel path-following method, in the presence of a static and a moving obstacle.

considering static and moving obstacles, as shown in Figs. 10a and 10b respectively. Based on simulation results, our RRT-based navigation method showed evidences of being a promising approach for real world application.

V. Conclusion and Future Work

In this paper we presented a navigation system to be deployed in automated vehicles for autonomous navigation in dynamic environments. Solutions to the problems of path-following and local path planning are proposed and simulation results are reported. Experiments have provided a comparison of path-following controllers, having our controller achieved interesting results and evidence of being a promising approach for real world application. Moreover, the RRT-based dynamic path-planning approach presented in this work provided evidence, in simulation, to be a potential solution in environments with static and moving obstacles. As future work we plan to extend our study to include an object classification process, intended to estimate object's categories in order to allow machine decision making according to specific safety requirements in obstacle avoidance. Moreover, experiments using a robotic platform, such as the ISRobotCar [23], in a real-world environment constitute part of our ongoing research.

Acknowledgments

This work is supported in part by the Portuguese Foundation for Science and Technology (FCT), under grant PTDC/EEA-AUT/113818/2009. Luis Garrote is supported by FCT under grant SFRH/BD/88459/2012.

References

[1] J. Schwendner, S. Joyeux, and F. Kirchner, "Using embodied data for localization and mapping," *Journal of Field Robotics*, vol. 31, no. 2, pp. 263–295, 2014.

[2] R. Solea and U. Nunes, "Trajectory planning and sliding-mode control based trajectory-tracking for cybercars," *Integrated Computer-Aided Engineering*, vol. 14, no. 1, pp. 33–47, 2007.

[3] D. Castro, U. Nunes, and A. Ruano, "Reactive local navigation," in *IEEE IECON 02*, vol. 3, Nov 2002, pp. 2427–2432 vol.3.

[4] S. Thrun *et al.*, "Stanley: The robot that won the DARPA grand challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[5] O. Amidi, "Integrated mobile robot control," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep., May 1990.

[6] M. Silva, L. Garrote, F. Moita, M. Martins, and U. Nunes, "Autonomous electric vehicle: Steering and path-following control systems," in *MELECON, IEEE*, March 2012, pp. 442–445.

[7] J. E. Naranjo, M. A. Sotelo, C. Gonzalez, R. Garcia, and T. de Pedro, "Using fuzzy logic in automated vehicle control," *Intelligent Systems, IEEE*, vol. 22, no. 1, pp. 36–45, 2007.

[8] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., 1998.

[9] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, June 2011.

[10] L. Jaillet, J. Hoffman, J. van den Berg, P. Abbeel, J. Porta, and K. Goldberg, "EG-RRT: Environment-guided random trees for kinodynamic motion planning with uncertainty and obstacles," in *IROS*, Sept 2011, pp. 2646–2652.

[11] K. Macek, M. Becked, and R. Siegwart, "Motion planning for car-like vehicles in dynamic urban scenarios," in *IROS*, Oct 2006.

[12] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT*," in *ICRA*, Shanghai, China, 2011, pp. 1478–1483.

[13] D. Geronimo, A. Lopez, A. Sappa, and T. Graf, "Survey of pedestrian detection for advanced driver assistance systems," *IEEE Tran. on Pattern Analysis and Machine Intelligence*, vol. 32, no. 7, pp. 1239–1258, July 2010.

[14] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *IEEE Tran. on Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, pp. 743–761, April 2012.

[15] D. Olmeda, C. Premebida, U. Nunes, J. M. Armingol, and A. de la Escalera, "Pedestrian detection in far infrared images," *Integr. Comput.-Aided Eng.*, vol. 20, no. 4, pp. 347–360, Oct. 2013.

[16] C. Premebida and U. Nunes, "Segmentation and geometric primitives extraction from 2D laser range data for mobile robot applications," in *Proc. 5th National Festival of Robotics, Scientific Meeting (ROBOTICA)*, Coimbra, Portugal, 2005.

[17] G. A. Borges and M. J. Aldon, "Line extraction in 2d range images for mobile robotics," *Journal of Intelligent & Robotic Systems*, vol. 40, no. 3, pp. 267–297, 2004.

[18] J. E. Guivant, F. R. Masson, and E. M. Nebot, "Simultaneous localization and map building using natural features and absolute information," *Robotics and Autonomous Systems*, vol. 40, no. 2-3, pp. 79 – 90, 2002.

[19] Y. Bar-Shalom and X. Li, *Multitarget-Multisensor Tracking: Principles and Techniques*. YBS Publishing, 1995.

[20] D. Ferguson , T. Howard, and M. Likhachev, "Motion planning in urban environments: Part I & II," in *IROS*, September 2008.

[21] J.-P. Laumond, *Robot Motion Planning and Control*. Berlin: Springer-Verlag, 1998.

[22] M.-P. Dubuisson and A. Jain, "A modified hausdorff distance for object matching," in *Pattern Recognition, Computer Vision and Image Processing*, vol. 1, Oct 1994, pp. 566–568 vol.1.

[23] M. Silva, F. Moita, U. Nunes, L. Garrote, H. Faria, and J. Ruivo, "ISRobotCar: The autonomous electric vehicle project," in *IROS*, Oct 2012, pp. 4233–4234, (Video Paper).