International Journal of Advanced Robotic Systems

OPEN ACCESS ARTICLE

Efficient Reactive Navigation with Exact Collision Determination for 3D Robot Shapes

Regular Paper

Mariano Jaimez[1,2]*, Jose-Luis Blanco[3] and Javier Gonzalez-Jimenez[1]

1 Department of System Engineering and Automation, University of Málaga, Spain
2 Department of Computer Science, Technical University of Munich, Germany
3 Department of Engineering, University of Almería, Spain
*Corresponding author(s) E-mail: marianojt@uma.es

Received 25 July 2014; Accepted 02 March 2015

DOI: 10.5772/60563

Abstract

This paper presents a reactive navigator for wheeled mobile robots moving on a flat surface which takes into account both the actual 3D shape of the robot and the 3D surrounding obstacles. The robot volume is modelled by a number of prisms consecutive in height, and the detected obstacles, which can be provided by different kinds of range sensor, are segmented into these heights. Then, the reactive navigation problem is tackled by a number of concurrent 2D navigators, one for each prism, which are consistently and efficiently combined to yield an overall solution. Our proposal for each 2D navigator is based on the concept of the "Parameterized Trajectory Generator" which models the robot shape as a polygon and embeds its kinematic constraints into different motion models.

Extensive testing has been conducted in office-like and real house environments, covering a total distance of 18.5 km, to demonstrate the reliability and effectiveness of the proposed method. Moreover, additional experiments are performed to highlight the advantages of a 3D-aware reactive navigator. The implemented code is available under an open-source licence.

Keywords Reactive Navigation, Mobile Robotics, Robot Motion, Obstacle Avoidance

1. Introduction

Reactive navigation is a crucial component of almost any mobile robot. It is one of two halves which, together with the path-planner, make up a navigation system according to the commonly used "hybrid architecture"[1]. Within this scheme, a reactive navigator works at the low-level layer to guarantee safe and agile motions based on real-time sensor data.

Traditionally, due to the lack of affordable 3D sensors and the limited computational resources available, reactive navigators have relied on two strong assumptions:

- The world is considered 2D. Since robots usually move on a flat surface, this implies that the third dimension (height) is ignored.
- The robot shape is simplified by a polygon or circle projected onto the 2D world.

These two simplifications force the reactive algorithm to adopt the worst-case scenario, that is, to work with the most restrictive section of the robot and the nearest obstacle detected in each direction. This limitation can complicate or even impede many robotic platforms from carrying out their tasks. In the most general case, the 2D reduction over-constrains the robot motion, marking as unfeasible some paths through which the robot could actually navigate. This effect arises when the robot does not have a constant vertical section, and it is particularly detrimental for robotic platforms equipped with a manipulator. In this specific case, any purely 2D approach would not allow the robot to place any object on any horizontal surface (e.g., a table) because the robotic arm and the surface would be superimposed in a 2D projection and would represent a collision for the 2D navigator. With the recent emergence of 3D range cameras and the current computational resources on board robots, these assumptions are no longer justified.

In this work we address the problem of planar navigation in indoor environments by a reactive navigation system which considers both the 3D shape of the robot and the 3D geometry of the environment (Figure 1). The proposed reactive navigator is based on the concept of "Parameterized Trajectory Generator" or PTG [2], a robust and effective 2D reactive navigator that models the robot shape as a polygon and embeds its kinematic constraints into different motion models. The contribution of this paper consists of extending such work to overcome the limitation of modelling the robot in 2D:

- The robot volume is now modelled by a number of prisms consecutive in height. Collisions are evaluated considering those exact prisms, including predicted robot orientations according to a number of path families, unlike many existing approaches which take the conservative "circular-robot approximation" and only consider circular paths.
- 3D obstacles coming from an arbitrary number of range sensors can feed the reactive navigator.
- The new 3D information is merged consistently and efficiently to yield an overall solution.

This generalization, called 3D-PTG navigator, has been extensively tested in varied and challenging scenarios. Two different robots, Giraff (see Figure 4 and Figure 5) and Rhodon (see Figure 5 and Figure 12), equipped with radial laser scanners and RGB-D cameras, have been chosen to conduct the experiments and demonstrate the potential of our approach. Overall, more than 20 hours of navigation are analysed, during which the robots covered a distance of 18.5 km in both office-like and house-like scenarios.

This paper is divided into eight sections. Section 2 describes the state of the art in reactive navigation. A brief summary of the 2D-PTG navigator is presented in Section 3. Its generalization to the 3D world is described in Section 4, and all the algorithm steps are explained in Section 5. Implementation details are given in Section 6 and the experiments are presented and analysed in Section 7. Results are divided into four subsections: two of them are intended to demonstrate the robustness of our approach, while the other two show the advantages of a 3D navigator as against the classic 2D approach. Finally, conclusions are discussed in Section 8.

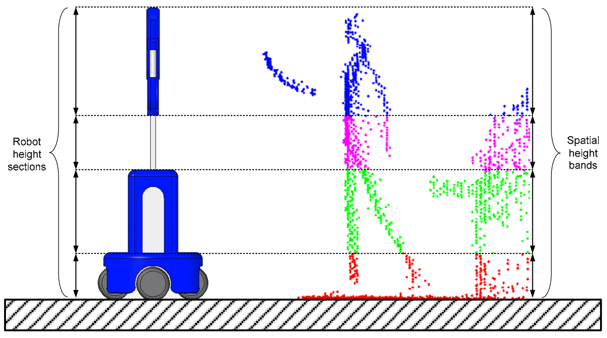

Figure 1. 3D obstacles are sorted in height bands according to the 3D shape of the robot

The implemented code has been added to MRPT [3] and is available under an open-source licence. Two demonstration videos of our approach, together with the code, can be found here: **http://mapir.isa.uma.es/mjaimez**

2. Related work

The first example of reactive navigation was probably given by the tortoises of Walter [4]. From then on, reactive navigation has been well studied, and some authors like Brooks [5] have defined it as the lowest hierarchical layer of a robotic motion control. The first successful methods, such as VFF [6], VFH [7] and VO [8], enabled robots to advance toward a given target while avoiding the obstacles encountered along the path, but they ignored both the robot shape and its kinematic constraints. Afterwards, reactive algorithms started to overcome these simplifications, solving the navigation problem in a velocity space where kinematic and dynamic constraints can be easily considered (typically, speed and acceleration limits). Within this category, Simmons [9] proposed to compute the optimal motion command in a curvature-velocity space where translational and rotational velocities are represented independently. In a similar way, the "Dynamic Window Approach" (DWA) [10], which was arguably the most successful strategy of this kind, minimizes an energy function to obtain the best motion command regarding the reachable obstacles and velocities within a short time interval. DWA is still in use today thanks to its implementation as a local planner in the popular Robot Operative System (ROS) "navigation" stack. These two approaches [9, 10] impose the feasible trajectory to be composed of circular arcs, so non-holonomic restrictions are also regarded, although, on the other hand, both the robot and the obstacles are still supposed to be circular. This circular shape assumption is also made in the extension of the Velocity Obstacle method [11].

Later improved solutions incorporated the robot shape into the reactive navigator. Minguez and Montano [12] defined the Ego-Kinematic Transformation (EKT): a mathematical procedure to transform the 3D configuration space into a new 2D space which implicitly contains the robot shape and its non-holonomic constraints. In this reduced space, the robot is a free-flying point and any holonomic obstacle avoidance method can be used to compute the solution. However, this approach still has a shortcoming: only circular paths are considered. This methodology was extended by Blanco et al. [2] with the generalization of path models through a novel approach called "Parameterized Trajectory Generator" (PTG). With this tool, several customized path models can be used in the reactive navigator and, at each iteration, the best one is selected according to some specific criteria, such as the collision-free distance for the selected movement, the minimum distance from the path to the target, etc. It must be noted that the robot becoming a free-flying point in this space comes at the cost of having a different set of obstacles in the transformed space, even for real stationary obstacles. However, as will be seen experimentally, this obstacle transformation can be made extremely efficient by means of precomputed look-up tables.

In this context, reactive navigators were effective enough, but they still had to assume that the world was 2D. Nonetheless, the improvement and availability of 3D range sensor during the last few years have made it possible to realistically tackle the problem of navigating in 3D environments. Most of the new approaches are based on processing 3D point clouds and use the resulting information to execute a 2D navigator. For example, the solution proposed by Surmann et al. [13] consists in scanning the environment with a tilting laser and extracting some semantic information (planes) which is projected onto the floor plane and utilized by a 2D navigator. Holz et al. [14] also use a tilting laser to generate 3D point clouds, which are processed to obtain the "2D Obstacle Map" and "2D structure Map". The former contains the minimum distance in each scan direction (i.e., closest obstacles) and is exploited by the reactive navigator, while the 2D Structure Map contains the maximum distance in each scan direction (i.e., furthest obstacles) which is likely to correspond to the environmental bounds and is used for robot localization. In contrast, Marder-Eppstein et al. [15] proposed to store the 3D information in a voxel grid: a 3D occupancy grid whose cells are marked as occupied, free or unknown. The robot navigation is controlled by two modules: the "global planner" and the "local planner". The "global planner" creates a high-level plan for the robot to reach the goal location and the "local planner" is a reactive navigator based on the aforementioned DWA [10]. No information is given about how the 3D voxel grid is interpreted by the 2D reactive navigator. Finally, the work recently proposed by Gonzalez-Jimenez et al. [16] addresses the problem of adding the 3D information provided by an RGB-D camera to a reactive navigator which was designed to work with radial laser scanners. To this end, they propose to adapt the Kinect depth image into a virtual 2D scan which, in turn, encapsulates the 3D world information.

From a different point of view, 3D navigation is also studied for legged robots and humanoids [17, 18, 19]. In these cases, point clouds are always analysed to extract semantic information, which is more convenient for the gait of this type of robot. In general, a 3D representation of the world has proved to be advantageous in many other aspects of the robot navigation, e.g., in localization [20].

3. Reactive navigation based on PTGs

For the sake of completeness, this section summarizes the PTG-based reactive navigator upon which our proposal for dealing with a 3D world is built. More details can be found in [2]. The PTG-based reactive navigator is based on a mathematical transformation that reduces the dimensionality of the Configuration Space (C-Space) [21] from 3D (x, y, ϕ) to 2D, incorporating in the transformation the geometrical and kinematical constraints of the robot. The robot thus becomes a free-flying point over this 2D manifold embedded in C-Space, and the collision avoidance problem is easier and faster to solve. This dimensional reduction is accomplished by restricting the robot motion to one of a set of parametric path models which are compliant with the robot kinematics (e.g., circular paths, as shown in Figure 2). The set of all possible robot poses according to any path model constitutes a 2D manifold ("sampling surface") embedded in the general C-Space (3D). The basic idea behind PTG-based navigation is to map those manifolds by means of 2D "Trajectory Parameter Spaces", or TP-Spaces. The mathematical transformation between the TP-Space and the C-Space is formulated by a "Parameterized Trajectory Generator" or PTG, a smooth mapping of TP-Space points into C-Space poses according to a certain path model. TP-Spaces are expressed in polar coordinates, where the angle α corresponds to an individual path from the family and the radius d indicates the normalized distance travelled along that path (Figure 2). The region of interest in a TP-Space is the circle of unit radius, that is, the sub-space $A \times D \subset \mathbb{R}^2$, where $A = \{ \alpha \mid \alpha \in [-\pi, \pi] \}$ and $D = \{ d \mid d \in [0, 1] \}$. Therefore, kinematically compliant paths become straight lines in TP-Space and the robot motion can be guided with simple holonomic methods like VFF [6] or ND [22], disregarding the robot shape and non-holonomic constraints.

Since several path models are included in the reactive system, several TP-Spaces are built. The mathematical transformation between the C-Space and the TP-Spaces is done by the inverse PTG function, defined as:

$$\begin{gathered} PTG^{-1} : Sampling\,surface \subset \mathrm{R}^2 \times S^1 \to A \times D \subset \mathrm{R}^2 \\ \{(x, y), \phi\} \to (\alpha, d) \end{gathered} \tag{1}$$

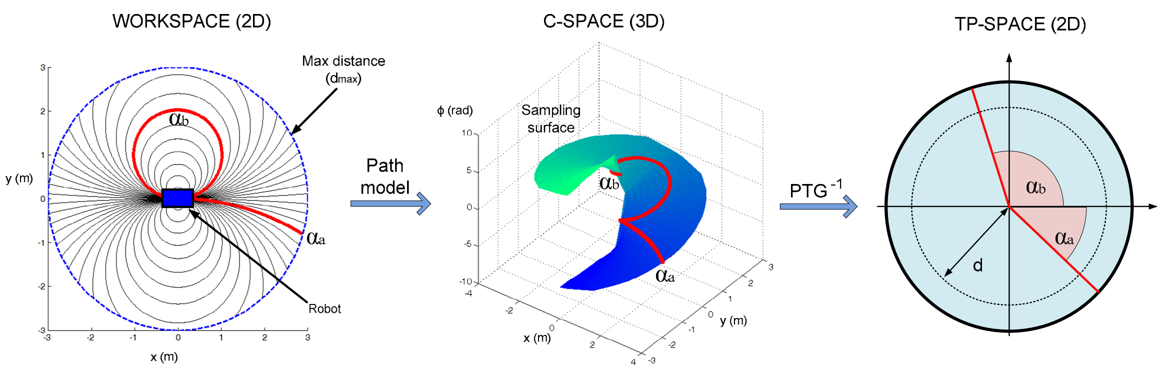

Figure 2. Mathematical transformations from the Workspace (2D) to the C-Space (3D) and then to the TP-Space (2D). Individual trajectories (α_a and α_b) form curves in the C-Space and are segments for a particular referenced angle in the TP-Space.

Equation (1) transforms both obstacle points and the target from the C-Space to the corresponding TP-Space, using path models such as those shown in Figure 3. In the resulting TP-Spaces, the robot becomes a free-flying point because its shape and its kinematic constraints are embedded into the mathematical transformation and, therefore, any holonomic method can be applied to get the best path α_b^i in the *i*-th TP-Space. All the α_b^i are subsequently evaluated and compared following some heuristic criteria which take into account the collision-free distance for the selected movement, the minimum distance from the path to the target, whether the robot will be heading to the target or not, etc. The result of this process will be the most suitable movement α_b given the obstacles, the target and the path models being used. Finally, the speed commands associated with α_b are calculated and sent to the robot. This sequence is repeated at a given frequency, typically higher than 20 Hz, such that the robot can move smoothly. In summary, the PTG-based reactive navigator has two inputs – the target relative pose and sensor data – and generates one output: the velocity command for the robot. The sequence of steps to compute the velocity commands is as follows:

1. For each path model, using its corresponding PTG, transform the obstacles and the target to the associated TP-Space.
2. For each path model, apply a holonomic reactive method to get the best path α_b^i in the TP-Space.
3. Select the best path α_b among the candidates α_b^i obtained from the different TP-Spaces.
4. Compute the linear and angular velocities and send them to the robot motor unit.

4. PTG-based reactive navigation in a 3D world

The main limitation of a 2D navigator, such as the one described above or any of those reviewed in Section 2, is that both the robot and the world are assumed to be 2D, that is, the robot section is considered to be constant and the detected obstacles are projected onto the floor plane, even if they are provided as 3D data by sensors like RGB-D cameras or lidars. This happens to be a valid simplification under the assumption that the robot has roughly the same horizontal profile all the way from bottom to top. However, many wheeled robots have a non-constant section (please visit **ROS – robots** to find many examples like *PR2*, *Gostai Jazz*, *Amigo*, etc.), in which case the 2D solution is suboptimal for two reasons: it takes the biggest section of the robot and also the closest obstacles regardless of their height position in space. Figure 4 illustrates this limitation with an obstacle configuration where a 2D reactive navigator would fail even though the robot has enough space to pass through. This kind of situation is quite common in cluttered environments and demands the addition of the third dimension (height) to the reactive navigator in order to successfully cope with it. For that purpose, we model the robot geometry through a set of prisms circumscribing the robot volume, as shown in Figure 5. Besides defining the 3D shape of the robot, it is also necessary to include the height coordinates of the obstacles or, more specifically, to sort them into height bands according to the height sections used to model the robot (Figure 1). Therefore, we decompose the 3D reactive navigator into N 2D navigators, N being the number of height sections that model the robot geometry. Each 2D navigator comprises an individual robot section and the obstacles in its corresponding height band. In order to obtain an overall solution for the robot, we combine the results for all the 2D navigators, as will be described later.

At this point, it is necessary to give a brief explanation of how obstacles are transformed into TP-Obstacles and what they represent (Figure 6). In the Workspace, obstacles are always considered to be points. Focusing on a single obstacle (or point), and given its coordinates, the robot shape and its location, we can calculate all the poses (x, y, ϕ) in the C-Space which imply a collision between the robot and the obstacle. This set of poses forms a volume called C-Obstacle. In addition, it is useful to recall that in C-Space every path model is a sampling surface. Thus, TP-Obstacles are obtained by transforming the 3D intersection between C-Obstacles and the sampling surface to the TP-Space.

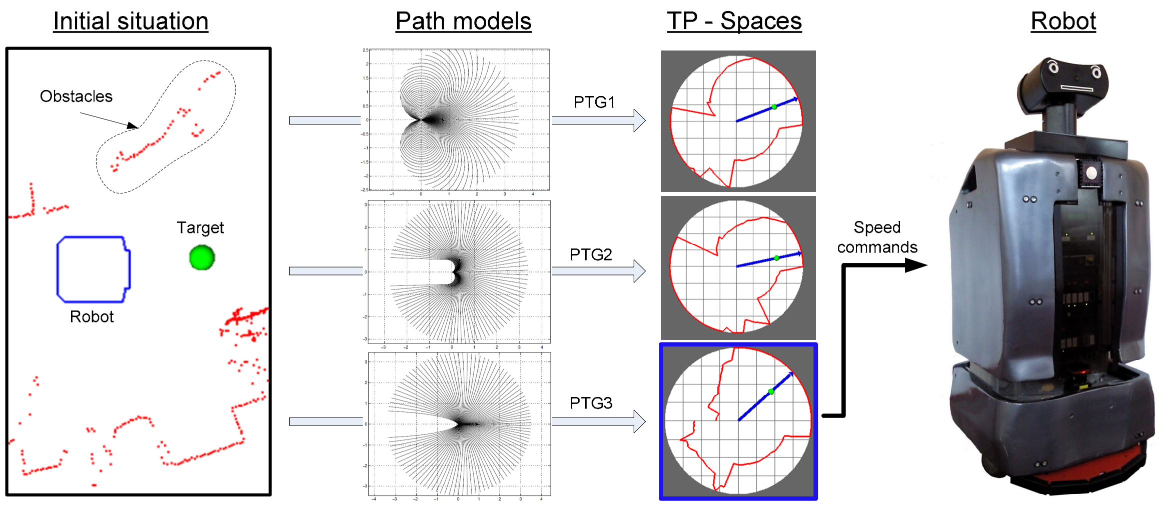

Figure 3. In this example three different path models are considered, leading to three different TP-Spaces. In both the Workspace and the TP-Spaces the target is marked as a green circle, while the obstacles are displayed in red. The best path for each TP-Space is shown with a blue arrow, and the best of all the three (PTG 3) is selected to provide the angular and linear robot velocities.

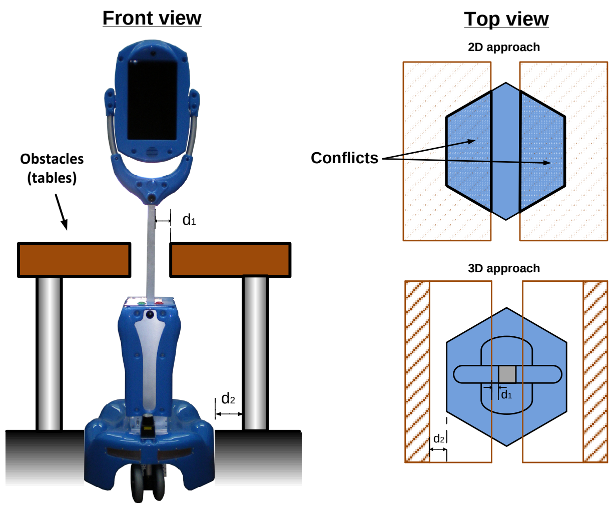

Figure 4. Example of how the 2D approach limits the reactive navigator performance

Formally, let $\sigma \in \mathbb{R}^2$ be a real obstacle in the Workspace, C-Obstacle(σ) its representation in C-Space and P a 3D point in C-Space. TP-Obstacles are defined as:

$$TP - Obstacle(\sigma) = \Big\{ (\alpha, d) \mid (\alpha, d) = PTG^{-1}(P), \\ \forall P \in C - Obstacle(\sigma) \cap Sampling\ surface \Big\} \quad (2)$$

Nevertheless, only the closest obstacle for each path α is relevant here as it marks the maximum distance the robot can travel along that path, always from the origin, without collision. The set of closest obstacles in TP-Space is called "TP-NavLimit". If there is no obstacle along the path α, its TP-NavLimit is set to 1 (the maximum distance in the normalized TP-Space), that is:

$$TP - NavLimit(\alpha) = \min\{1, d_m\} \quad (3)$$

where d_m is the minimum distance of the pairs $(\alpha,\ d) \in$ TP-Obstacles.

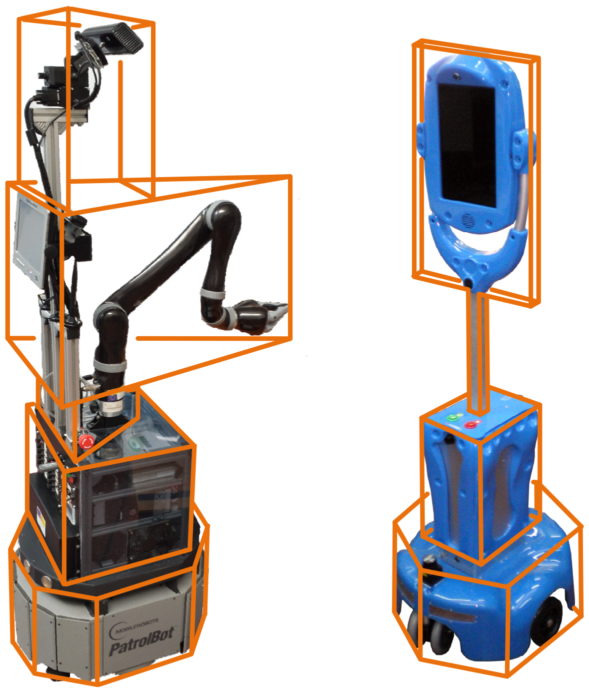

Figure 5. Example of 3D geometric models of the Giraff robot (right) and Rhodon (left)

Returning to the 3D reactive navigator, the same process is followed for the N height sections of the robot to obtain a TP-Space with N sets of TP-NavLimits, each indicating the maximum distance that the robot at that height section can travel along a given path model. Hence, the most restrictive TP-NavLimits are used to build the TP-Space for each α (Figure 7), that is:

$$TP - NavLimit(\alpha) = \min_{n} \{ TP - NavLimit_n(\alpha) \} \quad (4)$$

where $n = 1, 2 \ldots N$ are the height sections of the robot.

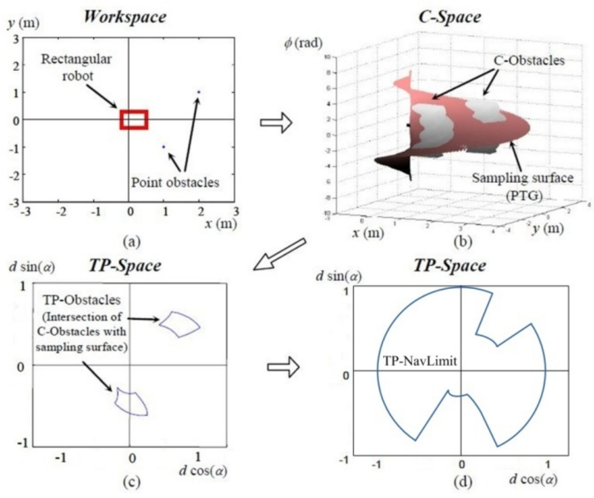

Figure 6. A set of obstacles in the Workspace is transformed to the C-Space and, according to a path family, to the associated TP-Space

We want to remark that calculating the minimum of all TP-NavLimits is not equivalent to projecting the most restrictive 3D obstacles onto the floor. Given that the TP-NavLimit of each individual robot section contains information about how far this part of the robot could travel in the 3D world according to some path models, the minimum of them shows how the whole robot could navigate in the same 3D world, since it encompasses the motion restrictions of every part of it.Thus, all these restrictions correspond to poses of the robot that would actually imply collisions with the environment, whereas the typical 2D obstacle projection is prone to creating motion constraints that do not correspond to any potential collision in the real world, hence over-constraining the robot motion.

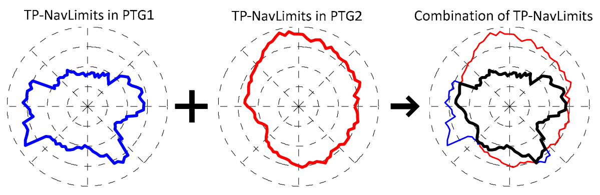

Figure 7. Example of combination of TP-NavLimits for a robot with two height sections. PTG1 is related to the first height section (blue) and PTG2 to the second one (red). The resulting TP-NavLimits (black) are calculated as the minimum of the TP-NavLimits associated with each individual PTG.

5. The 3D reactive navigation framework

In this section we describe the overall operation of the reactive navigator. It consists of a number of steps which are executed periodically at a given frequency (Figure 8). The inputs to the reactive system are the obstacles and the relative location of the target. Different kinds of sensors providing 3D obstacle points can be used simultaneously, typically laser scanners and RGB-D cameras. As previously mentioned, these 3D obstacle points are sorted according to the different height sections employed to model the robot volume.

Before applying the PTG transformations, a module called “Short-Term Memory” (STM) stores the position of close obstacles that might eventually become unseen by the robot sensors if they enter into their blind zone. This is particularly relevant for RGB-D cameras which have a narrow FOV and cannot detect obstacles at short distances. The STM module is implemented by N local occupancy grids centred at the robot pose onto which the 3D points within each slice in height are projected. The appropriate grid and cell sizes depend on the sensors used, the accuracy of the localization estimate, how cluttered the environment is, etc. The outputs of the STM block are sets of “virtual obstacles” whose coordinates are generated from those of the occupancy grid cells. These virtual obstacles are merged with the real ones coming from sensors and passed to the TP-Obstacle builder. Detailed information about the occupancy grids, their working principle and how they are implemented can be found in [16].

All the obstacles, real and virtual, are converted into TP-Obstacles for a number of path models and also for each height level. This results in $K \times N$ sets of TP-Obstacles and, subsequently, in $K \times N$ sets of TP-NavLimits. Then, the N sets of TP-NavLimits corresponding to each path model are combined, as explained in Section 4, yielding K TP-Spaces representing the robot navigability for each path model. Concurrently, the relative target location is also transformed to these TP-Spaces, where any holonomic method can be run, for example VFF [6] or ND [22], to get the most suitable path α_b^i for each path model i. The best path candidate α_b among all is then the one that maximizes an objective function that trades off several navigational criteria:

$$\alpha_b = \arg\max_{\alpha} \sum w_i f_i \left(\alpha_b^i \right) \quad (5)$$

with w_i being weighting coefficients and f_i factors which measure:

- f_1: The collision-free distance of each candidate (in TP-Space).
- f_2: The angular distance in the TP-Space between the target and the candidate.
- f_3: The minimum distance between the target and the path candidate.
- f_4: How different the new (tentative) and the previous speed commands would be (to soften the robot motion).

Finally, the linear and angular velocities are derived from α_b and sent to the robot motion control unit.

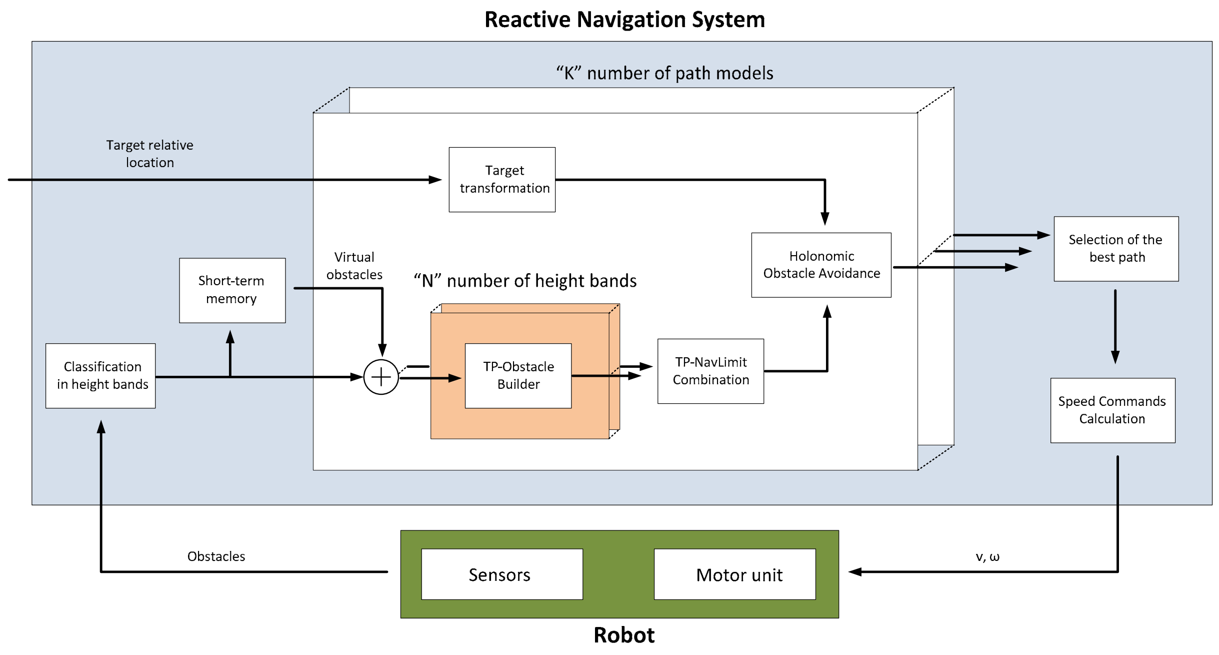
Reactive Navigation System
"K" number of path models
Target relative location
Target transformation
Holonomic Obstacle Avoidance
Selection of the best path
Short-term memory
Virtual obstacles
"N" number of height bands
Classification in height bands
TP-Obstacle Builder
TP-NavLimit Combination
Speed Commands Calculation
Obstacles
Sensors
Motor unit
v, ω
Robot

Figure 8. Scheme of the 3D reactive navigator with *K* different path models and *N* height sections

6. Experimental setup and implementation details

The 3D reactive navigator has been intensively tested for months in several scenarios to demonstrate its proper functioning. The Giraff and Rhodon mobile robots (Figure 5) have been chosen to conduct the experiments, as their heterogeneous profiles are appropriate to test the 3D reactive navigator's performance. In particular, the Giraff robot has been deployed and utilized for more than a year in several real apartments in Spain as part of the EU project GiraffPlus [23] using the proposed method to reactively navigate between nodes of a pre-established roadmap.

Giraff is a differential wheeled robot and has been equipped with a Hokuyo URG-04LX-UG01 laser and a PrimeSense Carmine 1.09 RGB-D camera, both facing forwards. Rhodon is a heavier differential wheeled robot equipped with two laser scanners (one Sick LMS200 and one Hokuyo UTM 30-LX, facing forwards and backwards respectively), and one Kinect camera placed at the top of the robot and tilted downward with an angle of 50 degrees. 3D points provided by all the range sensors are expressed with respect to the robot coordinate system and then merged before feeding the reactive navigator. In this preprocessing stage, the fused point cloud is downsampled, retaining only the most restrictive points at each height band. The frequency at which sensor data are read is adjustable; in our experiments it ranges from 10 Hz for the Hokuyo URG-04LX-UG01 (its maximum) to 30 Hz for the remaining sensors. Due to the lack of 3D sensory information at their back (Rhodon does contain a Hokuyo facing backwards but it is only used for localization), the robots are not allowed to move backwards during the experiments.

6.1 Configuration of the 3D reactive algorithm

First, we need to define the height sections that model the robot geometry (Figure 5). In these experiments we have modelled the Giraff robot with four consecutive prisms and Rhodon with five prisms (see Section 4 for further details). Second, path models and their characteristics have to be specified; in our case three different path models are considered (Figure 9): circular arcs, trajectories with asymptotical heading and trajectories with a minimal turning radius (see [2] for more details about their mathematical definition).

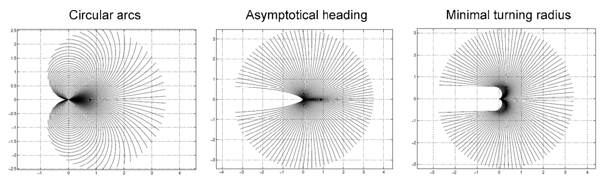

Figure 9. Set of path models considered in the experiments by the reactive navigator

Furthermore, the holonomic method can be chosen from two options: VFF [6] or ND [12], and their associated parameters can be customized too. We opted for the ND method because, in general, it outperforms the VFF. Finally, the reactive loop frequency is set to 20 Hz, which is fast enough to react to incoming sensor data without overloading the processor.

6.2 Speed regulation and recovery behaviour

Linear and angular velocity commands sent to the robot come from the PTG associated with the selected path, but they can be modulated or rescaled without violating the kinematic constraints of that path.

As a general rule, the robot should move carefully when it is surrounded by obstacles, but this is not always granted by the PTGs. For this reason, the speed commands are adjusted taking into account the proximity of obstacles, with the frontal obstacles having a greater influence than those at the sides. This speed regulation, which allows us to increase the average robot speed while keeping the navigation safe, can be adjusted depending on the robot dynamics and the desired balance between robot agility and conservativeness.

On the other hand, a basic recovery behaviour is implemented in case the robot gets stuck. Taking into account the fact that backward movements are not permitted, the reactive navigator assumes that the robot is trapped when both forward movements and rotations are impeded. In such situations, the robot starts to move backward slowly until it finds a feasible movement from the reactive navigator. If it is unable to find a way out after a few seconds, it stops (to avoid possible backward collisions) and keeps waiting until the environment changes or the user takes control of the situation.

7. Experiments

A wide variety of experiments have been conducted. The first two sets are intended to study the performance of the 3D navigator under circumstances that are habitual in many robotic applications: navigation in an office-like environment and navigation at home. To validate our proposal, the other two sets of experiments include some specific and demanding situations that cannot be addressed without 3D knowledge of the world. In all the experiments, localization relies on wheel odometry and laser scanners, which feed a particle filter implementation of localization based on a metric map of the environment [24]. Those geometric maps were previously built for each environment by means of a simple ICP-based incremental registration of laser scans.

7.1 Computational burden

In order to check the computational resources that the reactive navigator demands, we have tested how long one complete iteration of the reactive module takes on the Giraff robot, whose processor is an Intel i3 – 2310M 2.10 GHz with 4.0 GB of RAM. Considering four height sections and three different path models, the reactive iteration takes 4.8 milliseconds on a single CPU core, which implies a computational load inferior to 100 milliseconds per second for the 20 Hz implemented frequency. This leaves more than 90% of the CPU to the other robotic modules (localization, sensing, interface, etc.) which have to share the same computational resources.

We can compare this runtime with that of the 2D reactive navigator, i.e., considering only one height section to model the robot. In this case the reactive iteration takes 3.4 milliseconds, which implies that the 3D version is about 40% slower than the 2D for this particular configuration. The difference in time is not proportional to the number of height sections because, within one complete iteration of the 3D reactive approach, there are only a few steps that are executed for each height section of the robot model, as can be seen in Figure 8.

7.2 Navigating in an office-like environment

The Giraff robot has navigated autonomously around our lab floor for more than a year, mainly in an area which includes a long corridor, our two-room lab and the two contiguous labs. This scenario presents a wide variety of static and dynamic obstacles that the robot has to detect and dodge (Figure 10). Apart from the geometric map for localization, the robot is provided with a topological map from which navigational targets are generated randomly.

Table 1 shows the results of some of these navigational missions where navigational data were monitored to evaluate the reactive navigation performance. Overall, the robot has travelled 13.5 km with an average speed of 0.32 m/s and a top speed of 0.7 m/s. The incidents that took place during these sessions were classified into two categories. The first one, called "minor incidents", refers to smooth contacts or grazes that the robot itself can manage and solve autonomously without human intervention. The second category comprises those cases where the robot gets stuck and cannot resolve the situation by itself, needing human intervention.

Duration (s)	Distance travelled (m)	Average speed (m/s)	Minor incidents	Human intervention
5330	1771	0.332	0	0
1352	399	0.295	0	0
597	184	0.308	1	0
564	179	0.317	0	0
1500	479	0.319	1	0
1737	535	0.308	0	0
4749	1435	0.302	1	1
3974	1206	0.303	1	1
3764	1207	0.321	1	0
1614	537	0.333	2	0
5450	1811	0.332	4	0
4975	1587	0.319	5	1
1840	601	0.327	3	0
5228	1571	0.300	1	1
		Overall Results		
11.854 h	13.5 km	0.316	20	4

Table 1. Results of the experiments in an office-like environment

The incidents recorded are explained according to their nature:

- Minor incidents are due to wheel slippage and the relatively high response time of the whole system. Wheel slippage depends on the robot mechanics and the surface it is moving on, and causes the robot to move in an uncontrolled way. On the other hand, we have checked that the elapsed time from the moment an obstacle is detected until the time the robot starts to react to it is slightly higher than 0.5 seconds. This latency is the sum of a number of small response times, inertias, and communication delays between modules of the robotic architecture.
- Human intervention is mainly needed when the robot gets stuck due to unnoticed obstacles, most of them being small pieces or part of objects lying on the floor.

Figure 10 gives an idea of how the robot has been wandering during the experiments. The point map shown was built with a laser scanner and, hence, white areas may contain obstacles invisible to the laser scanner (tables, chairs or other objects) but not to the RGB-D camera. This explains why in the trajectory plot there are apparently free areas that the robot did not visit.

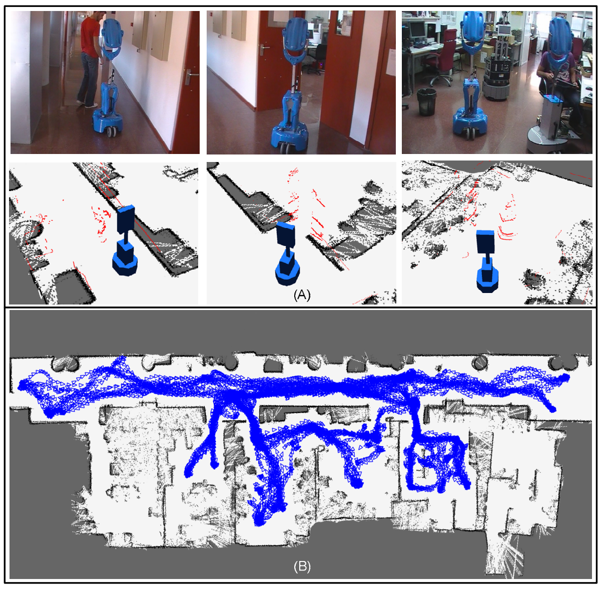

Figure 10. A) Some snapshots of the Giraff robot navigating in an office-like environment (first row) together with a virtual representation of the robot and the obstacles detected by the sensors onboard (second row). B) The trajectory described by the robot during one of the missions.

7.3 Navigating in a flat

As mentioned, the Giraff robot has been deployed for more than a year in several flats in Malaga (Spain) as part of the objectives of the EU project GiraffPlus [23]. The flat selected for the experiments has four rooms and presents a narrow navigable space with a reduced margin for manoeuvre (Figure 11). As a consequence, the Giraff maximum speed was lowered to 0.4 m/s in this case. The followed procedure is similar to that explained in the previous section: both metric and topological maps were built and provided to the robot which used them to navigate autonomously. Results are listed in Table 2.

We can observe that the average speed has decreased, which is not only a consequence of the maximum speed reduction, but is also caused by the many situations in which the robot performs a pure rotation to turn round, contributing zero to the average velocity (see Figure 11 B). The incidents that took place during these tests are explained following similar criteria to those mentioned in the analysis of the previous set of experiments:

- Minor incidents are mainly due to the high response time of the robot working loop, which becomes more relevant when moving in tight spaces.
- Only one human intervention was necessary because there were not many objects lying on the floor.

Duration (s)	Distance travelled (m)	Average speed (m/s)	Minor incidents	Human intervention
3003	448	0.149	2	0
4331	644	0.149	2	0
3845	571	0.149	1	1
4241	671	0.158	0	0
4263	654	0.153	1	0
4321	716	0.166	1	0
4384	731	0.167	3	0
3207	574	0.179	1	0
Overall Results				
8.776 h	5.01 km	0.159	11	1

Table 2. Results of the experiments in a four-room flat

7.4 Navigation with an outstretched robotic arm

During this experiment, Rhodon is commanded to perform a task that would be unfeasible using a 2D navigator. This task consists in visiting different desks with a robotic arm in a stretched position, emulating the process of collecting and delivering objects autonomously, but omitting the manipulation phase as it is outside the scope of this work. This is an illustrative example of a robotic application that necessarily requires 3D knowledge of the environment and the robot. As PTG-based navigation does not support changeable robot shapes, the arm is maintained at the same position during the whole navigation so that the same five prisms model the robot shape properly throughout this test (if the arm moved, its corresponding height section could be modelled according to the range of motion of the manipulator). We specify a blind pixel region for Kinect and neglect all the points observed at that region of the depth images since, given the camera pose on the robot (see Figure 5), the robotic arm is necessarily observed by the Kinect and would be considered an obstacle otherwise. Taking into account the weight (~ 50 kg) and height (1.8 m) of Rhodon, its maximum linear and angular speeds have been set to 0.4 m/s and 45 deg/s respectively. During the experiment, the robot has travelled 160 m around our lab visiting a total of seven different desks several times at random (Figure 12).

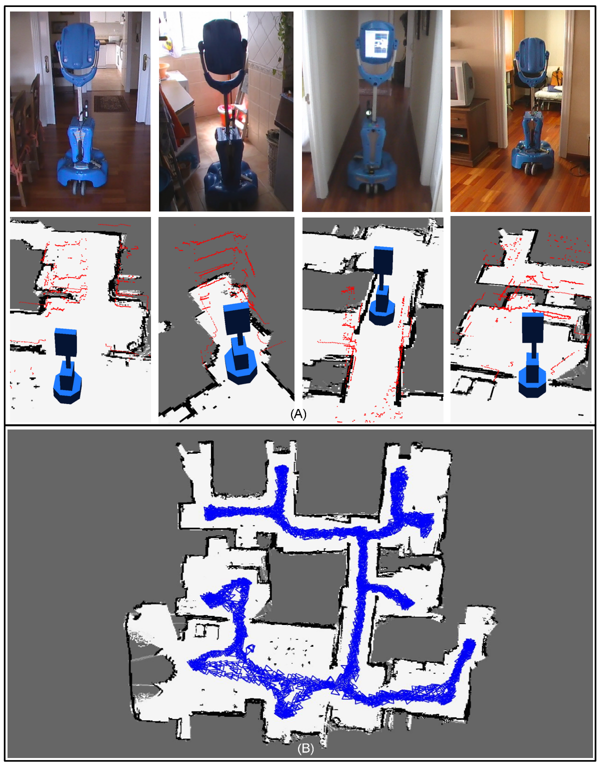

Figure 11. A) Some snapshots of the Giraff robot navigating in a flat (first row) together with a virtual representation of the robot and the obstacles detected by the sensors onboard (second row). B) The trajectory described by the robot during one of the missions. The geometric map was built by laser scans.

Aside from accomplishing its task, the 3D reactive navigator has shown an improved behaviour with respect to the 2D version since the robot was able, for example, to turn round when surrounded by chairs, tables or boxes while the arm was moving above them. No incidents occurred during this test. In Figure 12 we can observe the final pose of the robot when it reached some of its destinations and how the robotic arm lay above the corresponding desks.

7.5 Testing the reactive navigator's limits

A more extreme test to challenge the functioning of the 3D reactive navigator consisted of commanding the robot to go through a contour which has the same profile as the robot itself. For this purpose we cut out a piece of fabric and placed it at the door frame (Figure 13). The experiment was carried out placing the robot about 5 m away from the door at different positions and giving it a target outside of the lab. Sometimes we put an additional piece of fabric crossing the contour so as to check that the robot realized it could not go across it. The maximum velocity was set to 0.3 m/s and the short-term memory (STM) was not used (please see the demonstration video at the link attached in the introductory section).

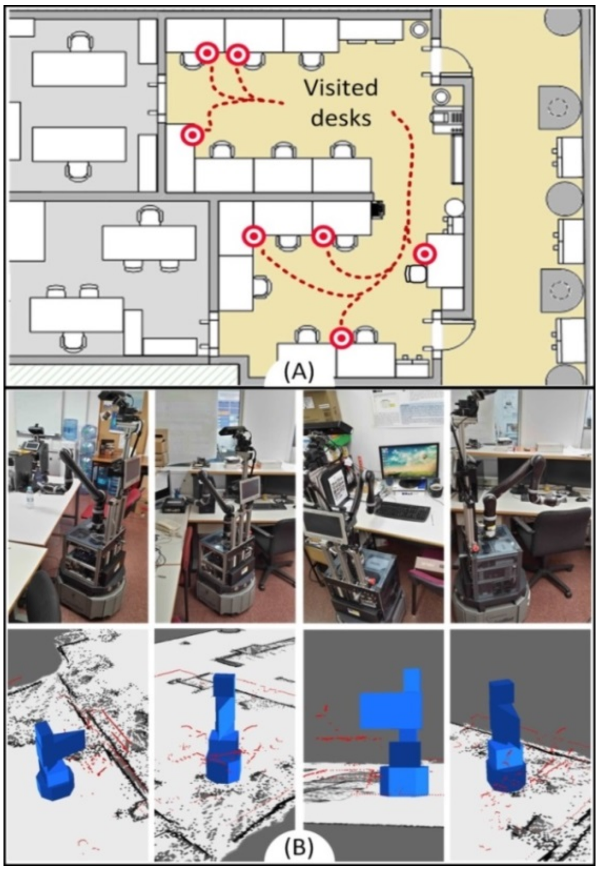

Figure 12. A) Schematic map of the lab and the desks that Rhodon visited during the experiment. B) Snapshots of Rhodon reaching its destinations or turning round surrounded by obstacles (first row) together with a virtual representation of the robot and the obstacles detected by the sensors onboard (second row).

We repeated this run 20 times and the robot always passed through the silhouette if the blocking piece of fabric was not present and always stopped otherwise. Quite frequently (about 50% of the times), however, the robot slightly touched the cloth as the field of view of the RGB-D camera is not wide enough to sense the whole clearance when the robot gets close to the door (Figure 13). We also made some trials activating the STM, but we found that using it had a counterproductive effect. As the localization module has a precision of few centimetres, virtual obstacles are inserted in the map carrying such positioning errors, which on occasion prevents the robot from seeing the clearance or cause undesired jittery behaviour in the reactive navigator. In this case, the errors in localization are higher than the spatial margins to pass through the clearance and, hence, the STM becomes useless. Nevertheless, the STM would be

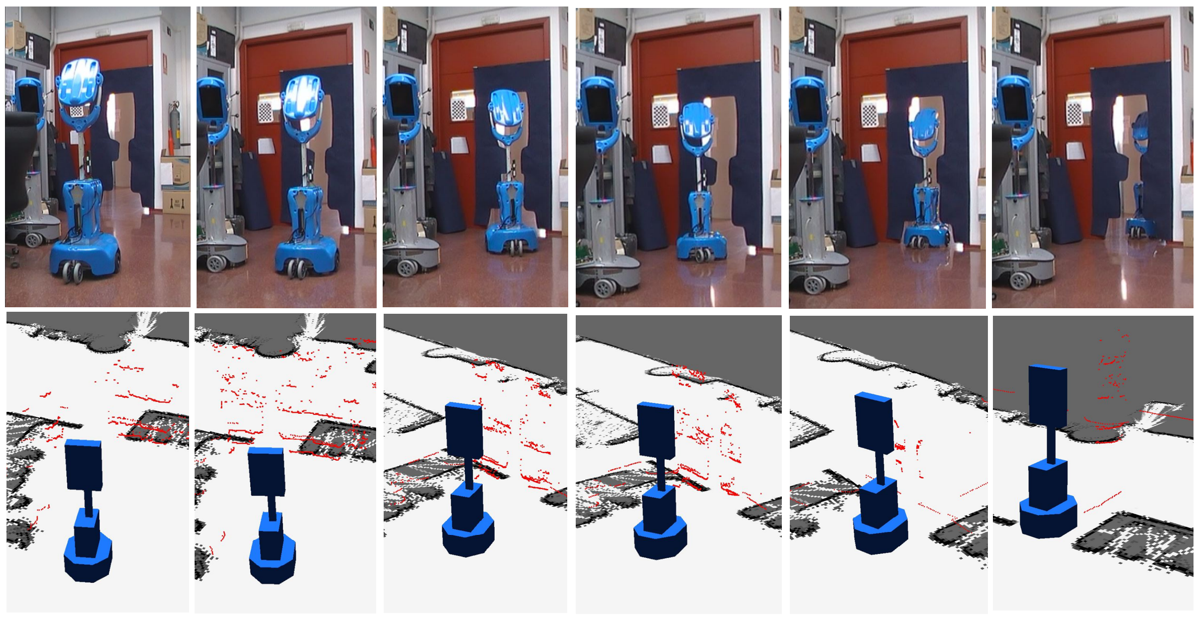
Figure 13. Real images (up) and representations of the robot, the detected obstacles and the map (down) showing how the robot goes through the contour

a good solution as long as the robot pose was estimated more precisely.

8. Conclusions

We have presented a 3D reactive navigator that can be adapted to almost any robot moving on a flat surface. We achieve high levels of versatility and manoeuvrability, as only very weak assumptions have been made to formulate this 3D approach, namely:

- The robot can be properly modelled in three dimensions as a set of prisms.
- Measurements coming from different kinds of sensor can be directly merged and read by the reactive navigator, provided they are expressed as 3D point sets.

The robot is allowed to move according to several kinematically compliant path models. Two robots with heterogeneous height sections were chosen to test the reactive navigator in different environments. A fair amount of experiments were conducted and the results support its proper functioning, although it may be conditioned by some factors. First, the robot mechanics has been shown to play an important role and can spoil the reactive navigator performance if the robot is not able to reproduce the motion commands quickly and accurately enough. Second, the coverage of the surroundings by the robot sensorial system is also a key factor that clearly delimits the quality of the reactive navigator. The number, type and placement of sensors needed for the robot to comprehensively sample its surroundings are aspects that require a great deal of attention. In this respect, we believe that active perception could be an effective solution to improve obstacle detection without demanding many sensorial resources. Although it has not been contemplated here, active perception represents a potential improvement and will be studied in future works.

9. Acknowledgements

This work has been funded by the Spanish Government under project DPI1011-25483, the EU FP7 Programme under the GiraffPlus project, and the Spanish grant programmes JDC-MICINN 2011 and FPI-MICINN 2012.

10. References

[1] Arkin R. (1999) Behavior-Based Robotics. Cambridge, USA: MIT Press.

[2] Blanco JL, Gonzalez-Jimenez J, Fernandez-Madrigal JA (2008) Extending obstacles avoidance methods through multiple parameter-space transformation. Autonomous Robots, vol. 24, no. 1, pp. 29-48.

[3] Blanco JL. The Mobile Robot Programming Toolkit (MRPT) [Internet]. Available from: http://www.mrpt.org, Accessed on 22 Dec 2014.

[4] Walter WG (1953) The Living Brain. New York: Norton.

[5] Brooks R (1986) A robust layered control system for a mobile robot. IEEE Journal of Robotics and Automation, vol. RA-2, pp. 14-23.

[6] Borenstein J, Koren Y. (1989) Real-time obstacle avoidance for fast mobile robots. IEEE Transactions on Systems, Man and Cybernetics, vol. 19, no. 5, pp. 1179-1187.

[7] Borenstein J, Koren Y (1991) The vector field histogram - fast obstacle avoidance for mobile

robots. IEEE Transactions on Robotics and Automation, vol. 7, no. 3, pp. 278-288.

[8] Tychonievich L et al. (1989) A maneuvering-board approach to path planning with moving obstacles. Proceedings of the 11th International Joint Conference on Artificial Intelligence, vol. 2, pp. 1017-1021.

[9] Simmons R (1996) The curvature-velocity method for local obstacle avoidance. IEEE International Conference on Robotics and Automation, Minneapolis, pp. 3375-3382.

[10] Fox D, Burgard W, Thrun S (1997) The dynamic window approach to collision avoidance. IEEE Robotics and Automation Magazine, vol. 4, no. 1, pp. 23-33.

[11] Fiorini P, Shiller Z (1998) Motion planning in dynamic environments using velocity obstacles. The International Journal of Robotics Research, vol. 17, no. 7,pp. 760-772.

[12] Minguez J, Montano L (2006) Abstracting vehicle shape and kinematics constraints from obstacle avoidance methods. Autonomous Robots, vol. 20, no. 1, pp. 43-59.

[13] Surmann H, Nuchter A, Hertzberg J (2003) An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments. Robotics and Autonomous Systems, vol. 45.

[14] Holz D, Lorken C,Surmann H (2008) Continuous 3D sensing for navigation and SLAM in cluttered and dynamic environments. Proceedings of the International Conference on Information Fusion, Cologne, Germany.

[15] Marder-Eppstein et al. (2010) The office marathon: Robust navigation in an indoor office environment. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).

[16] Gonzalez-Jimenez J, Ruiz-Sarmiento JR, Galindo C (2013) Improving 2D Reactive Navigators with Kinect. In 10th International Conference on Informatics in Control, Automation and Robotics (ICINCO), Reykjavik, Iceland.

[17] Morisset R et al. (2009) Leaving flatland: Toward real-time 3D navigation, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Kobe, Japan, pp. 12-17.

[18] Nishiwaki K, Chestnutt J, KagamiS (2012) Autonomous navigation of a humanoid robot over unknown rough terrain using a laser range sensor. The International Journal of Robotics Research, vol. 31, no. 11, pp. 1251-1262.

[19] Gutmann JS, Fukuchi M, Fujita M (2008) 3D perception and environment map generation for humanoid robot navigation. The International Journal of Robotics Research, vol. 27, no. 10, pp. 1117-1134, 2008.

[20] Kümmerle R et al. (2008) Monte Carlo localization in outdoor terrains using multilevel surface maps.Journal of Field Robotics, vol. 25, no. 6-7, pp. 346-359, 2008.

[21] Lozano-Pérez T (1987) A simple motion-planning algorithm for general robot manipulators. IEEE Journal of Robotics and Automation, vol. 3, no. 3, pp. 224-238.

[22] Minguez J, Montano L (2004) Nearness diagram (ND) navigation: Collision avoidance in troublesome scenarios. IEEE Transactions on Robotics and Automation, vol. 24, no. 1, pp. 45-59.

[23] Coradeschi S et al. (2013) Giraffplus: Combining social interaction and long term monitoring for promoting independent living. Proceedings of the 6th International Conference on Human System Interaction (HSI), pp. 578-585.

[24] Blanco JL, Gonzalez-Jimenez J, Fernandez-Madrigal JA (2010) Optimal filtering for non-parametric observation models: Applications to localization and SLAM. The International Journal of Robotics Research (IJRR), vol. 29, no. 14.