A ROS REACTIVE NAVIGATION SYSTEM FOR GROUND VEHICLES BASED ON TP-SPACE TRANSFORMATIONS

E. Rodríguez, J.L. Blanco, J.L. Torres, J.C. Moreno, A. Giménez, J.L. Guzmán
Universidad de Almería
Ctra. Sacramento 04120 Almería
enrique.romir@gmail.com, {jlblanco,jtm224,jcmoreno,agimfer,joguzman}@ual.es

Resumen

This work[1] *focuses on the analysis and benchmarking of the* `mrpt_navigation` *ROS package. The package includes a reactive navigation method based on Trajectory Parameter Space (TP-Space) transformations, together with other nodes such as an alternative particle-filter localization system. A description of its structure and methods in which it is based will be performed. To validate the suitability of the package, tests are performed with a mobile robot with strong kinematic constraints (Ackerman steering) and the result will be compared with the* `navigation` *ROS package, a standard for robots autonomous navigation.*

Palabras clave: Reactive navigation, planned navigation, obstacle avoidance, MRPT, ROS.

1 INTRODUCTION

Autonomous navigation is one of the most important objectives in mobile robotics. The most common package for navigation in the ROS environment is the `navigation` package[2] , which includes a localization system based on Adaptative Monte Carlo Localization (AMCL) approach and `move_base` for navigation itself. This package includes motion planning navigation algorithms with the drawback that, in general, are designed for circular shaped robots (e.g. obstacles are grown as if robots were circular) with differential configuration, so navigation becomes difficult for robots with other shapes or with different configurations (e.g. Ackerman), because in those cases recovery behaviors cannot be applied adequately or the robot cannot turn over itself.

As an alternative, in this work we will focus in describing and testing an alternative ROS navigation stack based on TP-Space transformations [1] and the MRPT library [2]. Mobile Robot Programming Toolkit (MRPT) is a cross-platform and open-source C++ library employed in common robotics research areas like Simultaneous Localization and Mapping (SLAM), computer vision, and motion planning (obstacle avoidance) [3]. The `mrpt_navigation` package [3] offers a realiable and effective alternative to address the problem of mobile robots navigation with kinematic constraints and any-shape robots.

The problem of mobile robots navigation has been arduously studied by different robotics communities and has two branches. On the one hand, motion planning navigation may be considered, in which a path is traced from the initial position to a goal given in a well known environment. Due to the computational cost of this approach, real time calculations are not ensured. On the other hand, we can deal with reactive or obstacle avoidance navigation, in which the path to goal is continuously changed in a partially-known or dynamic environment.

As a combination of both approaches, hybrid navigation arises from the calcualtion of the trajectories that are deformed for collisions avoidance in dynamic environments. Existing implementations in both branches have practical models that typically ignore the kinematic constraints of the robot, as its shape or configuration. The most used are based on treating the robot as a free-flight point that can perform straight or circular movements in any direction.

Therefore, as an improvement of these simplifications, space transformations methods [1, 4] were proposed as an alternative for reactive navigation to take into account the shape of the robot and its kinematics constraints in planar environment. This is achieved by finding the right movements to reach the goal avoiding collisions in a transformed space that embeds the robot shape. Reactive navigation is based on finding the free space around the robot for trace a motion path. This search is performed with some path models that measure the distance to obstacles. All existent reactive methods use path models that are an extension of the robot short-term motor actuation, where straight paths are employed for holonomic robots

[1] To appear in: XXXVII Jornadas de Automática, 2016.

[2] `http://wiki.ros.org/navigation`

[3] `http://wiki.ros.org/mrpt_navigation`

and circular arcs are considered for non holonomic ones.

The proposals in [1, 4] go beyond these models and implement more types of path models to help the robot finding more efficient and shorter paths, keeping in mind that it can not be an arbitrary path, it must meet the robot kinematic constraints.

The system allows decoupling kinematic constraints and obstacle avoidance using path models to transform the motion path and obstacles to a space of less complexity, called Trajectory Parameter Space (TP-Space). This transformation can treat the robot as a free-flying point in TP-Space since the kinematic constraints and dimensions of the robot are already embedded into the transformation process. In this way, we can entrust the obstacle avoidance task in the transformed space to any standard method for free-flight point robots. The generalization of path models is called Parameterized Trajectory Generator (PTG) in [1], allowing to implement the transformation of any path model and at the same time select the best alternative in every moment that suits in the motion plan, not only using circular path as traditional methods.

This paper relies in the use of other path models instead of circular arcs that allows the robot to find alternatives beyond simple obstacle avoidance and which are adapted to their kinematic constraints or shape. Furthermore, it could evaluate multiple path models simultaneously and select the most suitable at each step.

The rest of the paper is organized as follows. In section 2, TP-Space and PTGs are introduced. Section 3 discusses the alternative ROS package. In section 4, we will present a comparison between the two navigation stacks. Finally, conclusions are outlined in section 5.

2 BACKGROUND

Previous section mentioned the different navigation approaches. Next we explore in detail the basis of the TP-Space navigation package.

2.1 Trajectory Parameterized Space

Reactive navigation is based on transformation parameter space method, which consists of a frame where a robot with any shape and kinematically constraint becomes in real time in a free-flight point in a space where common obstacles avoidance methods can be applied.

For motion planning navigation is often used the Configuration Space (C-Space) [5], which represents the robot as a point at the cost of great complexity and the difficulty of using it accurately in real time.

A simplification of this space is the Velocity Space (V-Space) [6, 7, 8, 9, 10]. V-Space represents the space of linear and angular velocities of the robot, so goal points are curved paths in this space. The disadvantage of this is that the free-space sampling only can be performed with curved arcs, so other paths models can not be applied.

The Trajectory Parameterized Space (TP-Space) [4] arises as an alternative to the V-Space, built upon the following observation: while an arbitrary path is described in the three-dimensional C-Space (2D position plus heading), the poses along a circular arc can be defined in TP-Space through two parameters only, namely the path curvature and the distance along the arc. The majority of methods dealing with non-holonomic robots assume a family of compatible circular paths from which to select the robot movement at each instant of time. MRPT reactive navigation is based on this space where the robot can be treated as a free-flight point and assuming a decoupled kinematic constraints and the obstacle avoidance problems.

Hence, TP-Space is defined as a two-dimensional space where each point corresponds to a robot pose on C-Space sampling surface. The components of this space are an angular component α and a distance d.

2.2 Parameterized Trajectory Generator

We mentioned before that MRPT implements a set of path models to measure the distance to obstacles. These path models are called Parameterized Trajectory Generators (PTGs) and represent an allocation of TP-Space points (α,d) at C-Space positions ((x,y),φ) so the straight path from origin becomes compatible with C-Space.

Not every function that is designed is a valid PTG. They must ensure obstacle avoidance considering the kinematic constraints. Must meet the following objectives:

- It should generate consistent paths for reactive navigation because it uses a memoryless system.
- Regardless of heading, no more than one trajectory should exist taking the robot from its current pose to any other location.
- It must be continuous to ensure that transformations do not modify the topology of the robot planar workspace.

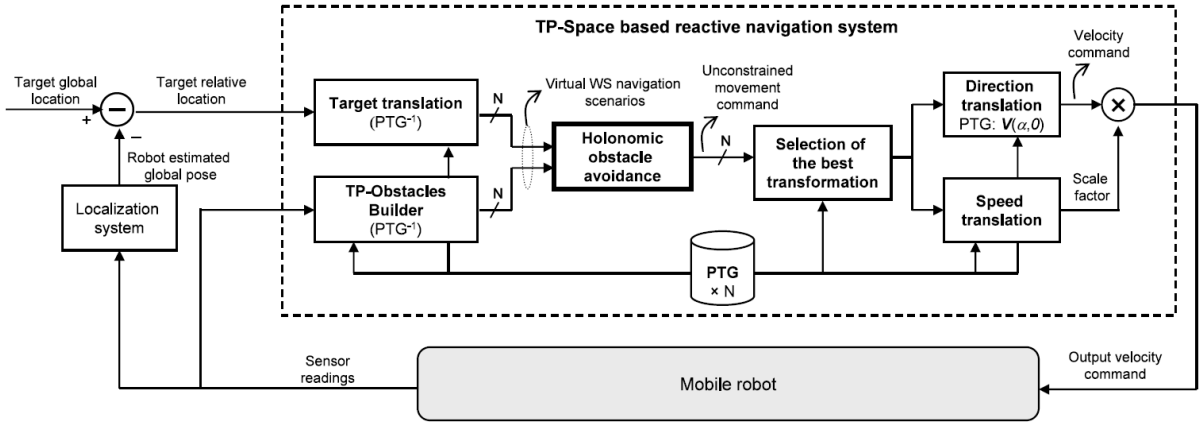


Figure 1: A complete reactive navigation system based on TP-Space involves translating obstacles and the target location into the TP-Space, through a variety of PTGs simultaneously. Each one generates a virtual WS navigation scenario which is solved by a simple obstacle avoidance method. Next, the resulting movements are evaluated to find the most advantageous transformation, which is selected to generate the real robot velocity command using the PTG design equations.

Next we briefly describe the PTGs implemented in MRPT and tested in this benchmark, which are summarized in table 1:

- **C PTG:** *Circular trajectories*

  This is the simplest path model and the common for traditional reactive navigation. Velocities remain constant along the path.

- **α-A PTG:** *Trajectories with asymptotically-known heading.*

  These trajectories are generated by linear and angular velocities, which are directly/inversely proportional to the difference between the robot heading and the parameter α.

- **α-SP PTG:** *Trajectories built upon a spiral segment*

  The purpose of this template is to show that valid and consistent reactive trajectories exist and which are not a composition of circular arcs and straight segments.

- **C|$\mathbf{C}_{\pi/2}$S and CS PTG:** *Trajectories built from sequences of straight and circular arc paths.*

  These paths represent optimal path models for car-like robots with a minimum turning radius R.

3 THE MRPT ROS NAVIGATION PACKAGE

Autonomous robot navigation is done commonly with the `navigation` ROS package that provides the localization node Adaptive Monte Carlo Localization (AMCL), the `move_base` controller and several planners.

This package is specially developed for robots with a differential configuration and a circular shape, so we must adapt de code for robot with other configuration or make our owns planners.

Alternatively to this package we find `mrpt_navigation`, a package that implements a localization node similar to AMCL (supporting additional sensors and map types) and a reactive navigation system (figure 1).

The *mrpt_navigation* package includes the following nodes:

- `mrpt_bridge`: C++ functions that convert between common ROS messages and MRPT classes.
- `mrpt_local_obstacles`: Construction of a local obstacle map (point cloud, voxel or occupancy grid) from recent sensor readings within a configurable time window.
- `mrpt_localization`: Node for 2D robot localization with a particle filter and many different kinds of metric maps. It is a wrapper for MRPT's particle filtering algorithms that can be seen as an alternative to ROS `navigation amcl`.
- `mrpt_map`: Node to publish metric maps (static, prebuilt).
- `mrpt_msgs`: Common messages for MRPT packages.

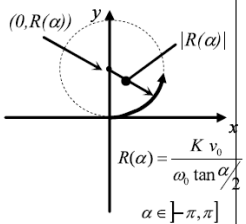
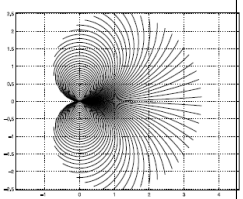
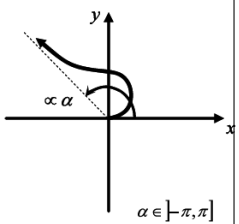
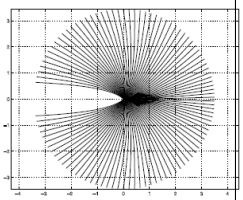
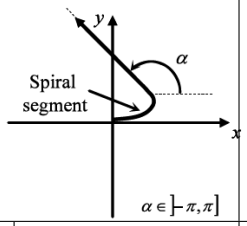
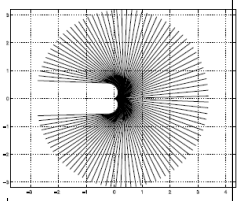
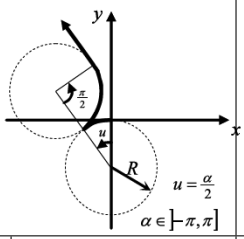
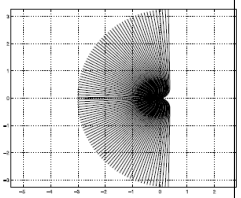
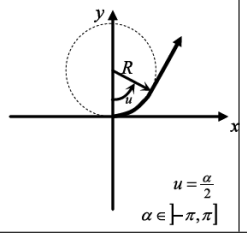
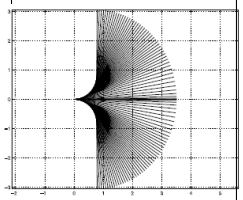
PTG	Type of trajectory	PTG equations	Design parameters	Example of generated paths
C		$V(\alpha,t)=\begin{bmatrix} Kv_0 \\ w_0\tan(\frac{\alpha}{2}) \end{bmatrix}$	v_0, w_0 $k=\pm 1$	
α-A		$V(\alpha,t)=\begin{bmatrix} v_0 e^{-\left(\frac{\alpha-\phi(\alpha,t)}{K_v}\right)^2} \\ w_0\left(\left(1-e^{-\left(\frac{\alpha-\phi(\alpha,t)}{K_w}\right)^2}\right)^{-1}-\frac{1}{2}\right) \end{bmatrix}$	$v_0, w_0,$ k_v, K_w	
α-SP		$V(\alpha,t)=\begin{bmatrix} f_v(\alpha,t) \\ f_\omega(\alpha,t) \end{bmatrix}\rightarrow\begin{cases} f_v(\alpha,t)=v_0\left(1-\beta\left(\frac{\alpha-\phi(\alpha,t)}{2\pi}\right)^2\right) \\ f_\omega(\alpha,t)\begin{cases}\omega_0, \alpha>\phi(\alpha,t)\\ 0, \alpha=\phi(\alpha,t)\\ -\omega_0, \alpha<\phi(\alpha,t)\end{cases}\end{cases}$	v_0, w_0, β	
C|$C_{\pi/2}$S		$V(\alpha,t)=\begin{cases} \begin{bmatrix} -v_0 \\ \frac{v_0}{R} \end{bmatrix}, t\leq\frac{R}{v_0}\frac{\lvert\alpha\rvert}{2} \\ \begin{bmatrix} v_0 \\ \frac{v_0}{R} \end{bmatrix}, \frac{R}{v_0}\frac{\lvert\alpha\rvert}{2}<t\leq\frac{R}{v_0}\left(\frac{\lvert\alpha\rvert}{2}+\frac{\pi}{2}\right) \\ \begin{bmatrix} v_0 \\ 0 \end{bmatrix}, \frac{R}{v_0}\left(\frac{\lvert\alpha\rvert}{2}+\frac{\pi}{2}\right)<t \end{cases}$	v_0, R	
CS		$V(\alpha,t)=\begin{cases} \begin{bmatrix} v_0 \\ \frac{v_0}{R} \end{bmatrix}, t\leq\frac{R}{v_0}\frac{\lvert\alpha\rvert}{2} \\ \begin{bmatrix} v_0 \\ 0 \end{bmatrix}, \frac{R}{v_0}\frac{\lvert\alpha\rvert}{2}<t \end{cases}$	v_0, R	

Table 1: Some examples of valid PTG design templates which have been evaluated on real robots. Columns represent the relationship of the parameter α with the type of trajectory, the design equations (the trajectory velocity vector), and a graphical example of generated paths in C-Space (as a 2D top view) for each approach. The employment of many other models increments the chance for the robot to find a good movement. Design parameters of the templates are v_0 and ω_0, the linear and angular maximum desired velocities, respectively, and R, the minimum turning radius for a car-like robot model in the two latest templates.

- `mrpt_rawlog`**:** Node for ROS topic logging, like rosbag but storing logs in an MRPT-specific file format called *rawlog*, compatible with the GUI application *RawLogViewer*.

- `mrpt_reactivenav2d:` Pure reactive navigation with TP-Space algorithms.

Using a plain-text configuration file, all navigation parameters as the dimension of the robot, the maximum velocities (linear and angular), a filter to change the speed gradually or the obstacle avoidance method (ND or VFF), are configured.

This reactive system implements two obstacles avoidance methods that can be applied in any situation:

- *Vector Force Field (VFF)*

  The main objective of an obstacle vector field is to produce a force which acts on the robots in such a way that it avoids the obstacle. This is most commonly done by creating a repulsive force which diminishes as the distance from the obstacle increases. In an instance where there are multiple obstacles multiple forces are added together to create one obstacle vector field. Both this vector field and the environmental vector field are superimposed and the result is used to determine the movement of the vehicle [11].

- *Nearness Diagram (ND)*

  By using space divisions, entities as nearby obstacles or free areas are identified and used to select among a discrete set of potential actions. In real time, the sensory information is used to identify one situation, and the associated action is executed computing the motion commands [12].

4 EXPERIMENTAL RESULTS

The objective of this work is to provide a preliminary benchmark of both navigation packages in a realistic situation, revealing the advantages and disadvantages offered by each one for a rectangular robot with Ackerman configuration.

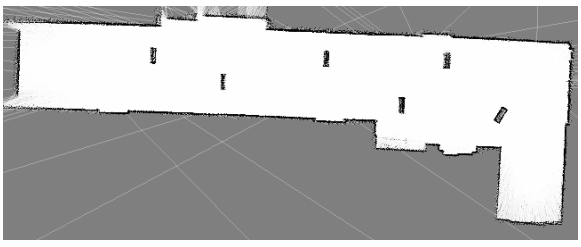


Figure 2: Map built for the experiments area.

The test is performed on a circuit where a number of obstacles have been placed and the robot has to overcome them. The sensorial system is based in a low-cost LIDAR and encoders. A representation of the map including the obstables is shown in figure 2. Time spent and trajectory to reach the goal will be evaluated as well as the behavior of the robot in speed and direction. Videos for one real experiment run with each method are available online[4].

4.1 Motion planning navigation

For motion planning navigation it has been used the `navigation` ROS package, with `"global_planner"` as global planner which generates a path from robot position to the goal, and `"dwa_local_planner"` as the local planner, tracing small paths in front of the robot that track the global trajectory.

In figure 3 it can be seen in red the trajectory that `global_plan` traces and the local trajectory that the robot tries to follow is represented in green.

Experimental tests have shown that the robot moves intermittently or "shaky", that is, the linear velocity changes constantly and the robot does not move continuously. Due to planning navigation, when the robot moves, trying to follow the `global_plan`, the servo-direction moves in both ways following the trajectory's line.

Figure 4 shows a representation of the linear and angular velocities of the robot during the test. The linear velocity is shown in blue and the angular velocity (rotational) is shown in red.

Due to robot hardware it has established a maximum linear speed of $0.7m/s$ and a maximum angular speed of $2rad/s$. These settings allow the robot to move with no significant alterations of the planning navigation as a consequence of sudden accelerations.

The robot completed the test in 96.66 seconds.

4.2 MRPT Reactive navigation

For reactive navigation it has been used the `mrpt_navigation` ROS package, with two PTGs: circular trajectories (C) and trajectories with asymptotical heading (α-A). For obstacle avoidance, the Vector Force Field method will be used.

In figure 5, it can be seen a representation of the linear and angular velocities, as in the motion planning navigation. In this case, the linear speed remains constant around $0.5m/s$ most of the time and the angular velocity change when the robot has to dodge an obstacle.

[4]Global planner `https://vimeo.com/133033742` and reactive method `https://vimeo.com/133034961`

Figure 3: Trajectory obtained with the motion planning approach.

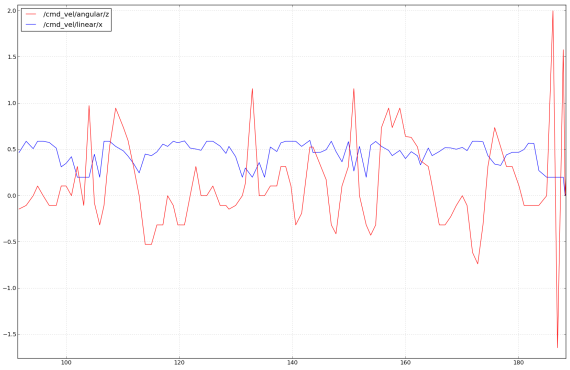


Figure 4: Velocities during the execution of the planned tracjectory.

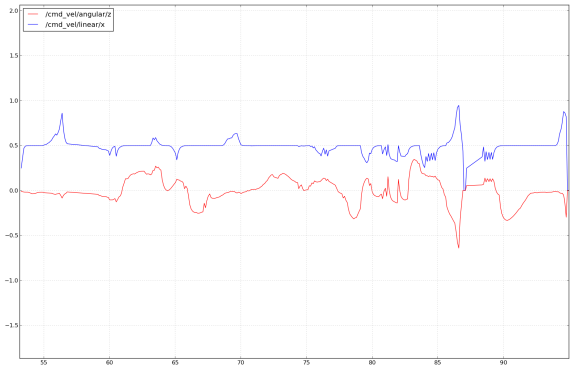


Figure 5: Velocities during the execution of the reactive navigation.

Figure 6 shows the path followed by the robot during the test. It can be seen that in the obstacle avoidance, the robot traces smooth curves around the obstacle and always maintains a continuous linear path. When the robot approaches an obstacle the linear speed decreases.

The robot completed the test in 41.63 seconds.

4.3 Localization

In both navigation packages a localization system based on an adaptive particle filter [13] has been used. For motion planning the `amcl` package included in ROS `navigation` has been used and for reactive navigation it has been used `mrpt_localization` package with a similar interface to `amcl` but supporting different particle-filter algorithms and sensors. In both cases, the estimated robot position in the map is shown as a cloud-point of red arrows, as depicted in figure 3.

5 CONCLUSIONS

The new `mrpt_navigation` ROS package has been presented and compared with respect to the well-known `navigation` ROS package through a series of experiments in real-world conditions with a medium-size robot. Both navigation systems affect the robot behavior differently. The trajectory traced for the robot in reactive navigation is sim-

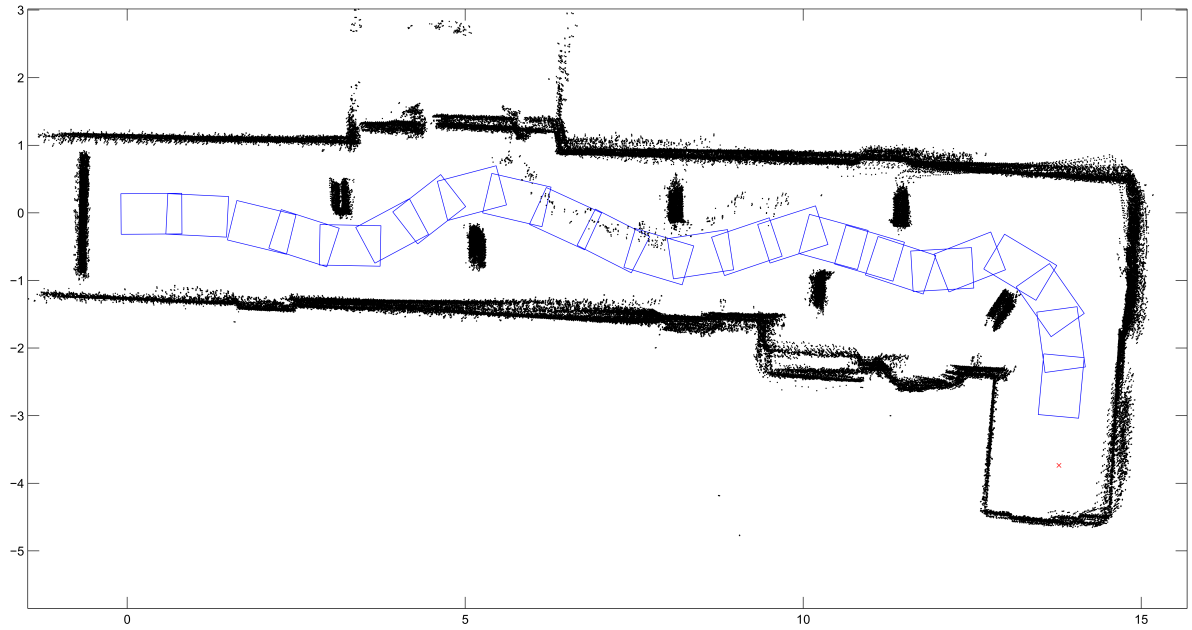


Figure 6: Trajectory obtained for the proposed reactive navigator ROS package. Note that the entire navigation is the result of one single target sent to the navigator at the beginning of the experiment.

ilar to the `global_plan` path calculation, keeping some distance from the obstacles. Instead of this similarity, the traveled path in the planning navigation has not been accurate to the calculated trajectory.

As for velocities, it is observed that the linear velocity which maintains the robot during reactive navigation is more constant than planning navigation as well as the continuous movement of the robot. Comparing the angular velocity it is also denoted that is smoother in reactive navigation than planning navigation and the small peaks that occur do not affect the robot trajectory. Finally, comparing the test run time, reactive navigation reaches the goal twice faster than planning navigation.

Findings from the results obtained were that planning navigation, despite reaching an optimum path to goal, strongly depend on a local navigator system that can follow the path accurately. Moreover, reactive navigation provides accurate and continuous navigation system but lacks a predetermined optimal path on a map. Reactive navigation does not ensure obtaining the shortest or most appropriate path to the goal, thus it should be better used in combination with a top-level planner.

Acknowledgments

This work has been funded by the National R+D+i Plan Project DPI2014-56364-C2-1-R of the Spanish Ministry of Economy and Competitiveness and ERDF funds.

References

[1] J.-L. Blanco, J. González-Jiménez, and J.-A. Fernández-Madrigal, "Extending obstacle avoidance methods through multiple parameter-space transformations," *Autonomous Robots*, vol. 24, no. 1, pp. 29–48, 2008.

[2] J. L. Blanco *et al.*, "Mobile robot programming toolkit (mrpt)."

[3] A. Harris and J. M. Conrad, "Survey of popular robotics simulators, frameworks, and toolkits," in *Southeastcon, 2011 Proceedings of IEEE*, pp. 243–249, IEEE, 2011.

[4] J. Minguez and L. Montano, "Abstracting vehicle shape and kinematics constraints from obstacle avoidance methods," *Autonomous Robots*, vol. 20, no. 1, pp. 43–59, 2006.

[5] Lozano-Pérez., "A simple motion-planning algorithm for general robot manipulators," *IEEE Journal of Robotics and Automation*, vol. 3, no. 3, pp. 224–238, 1987.

[6] K. O. Arras, J. Persson, N. Tomatis, and R. Siegwart, "Real-time obstacle avoidance for polygonal robots with a reduced dynamic windwow," *IEEE International conference on robotics and automation*, vol. 1, pp. 678–685, 2002.

[7] W. Feiten, R. Bauer, and G. Lawitzky, "Robust obstacle avoidance in unknown and cramped environments," *IEEE International conference on robotics and automation*, vol. 3, pp. 2412–2417, 1994.

[8] G. Ramirez and S. Zeghloul, "Collision-free path planning for non-holonomic mobile robots using a new obstacle representation in the velocity space," *Robotica*, vol. 19, pp. 543–555, 2001.

[9] C. Schlegel, "Fast local obstacle avoidance under kinematics and dynamic constraints for a mobile robot," *IEEE/RSJ International conference on intelligent robots and systems*, vol. 1, pp. 594–599, 1998.

[10] R. Simmons, "The curvature-velocity method for local obstacle avoidance," *IEEE International conference on robotics and automations*, vol. 4, pp. 3375–3382, 1996.

[11] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitation for mobile robot navigation," *IEEE International conference on robotics and automation*, vol. 1, pp. 1398–1405, 1991.

[12] J. Minguez and L. Montano, "Nearness diagram navigation: Collision avoidance in troublesome scenarios," *IEEE Transactions on robotics and automation*, vol. 20, no. 1, pp. 45–59, 2004.

[13] D. Fox, "Kld-sampling: Adaptive particle filters," in *Advances in neural information processing systems*, pp. 713–720, 2001.