The Proceedings of 2006 IEEE/RSJ International Conference on Robots and Intelligent Systems

Beijing, China
October 9-15, 2006

IEEE Catalog Number: 06CH37780D
ISBN: 1-4244-0259-X
Library of Congress: 2006921529

Proceedings of the 2006 IEEE/RSJ
International Conference on Intelligent Robots and Systems
October 9 - 15, 2006, Beijing, China

The Trajectory Parameter Space (TP-Space): A New Space Representation for Non-Holonomic Mobile Robot Reactive Navigation

Jose-Luis Blanco, Javier Gonzalez, Juan-Antonio Fernández-Madrigal
Dept. of System Engineering and Automation
University of Malaga
Málaga, Spain
{jlblanco,jgonzalez,jafma}@ctima.uma.es

***Abstract* – The reactive navigation of a non-holonomic mobile robot implies selecting at each instant of time a motion command satisfying two conditions: to avoid collisions and to comply with the robot non-holonomic constraints. Most proposed reactive navigation approaches deal with these requirements simultaneously in an indivisible way. This paper proposes a clear separation of these problems by introducing a representation space where a robot losses its kinematics restrictions and can be dealt as a "free-flying-point." The collision avoidance can therefore be solved by existing holonomic methods, which are able to steer non-holonomic, any-shaped robots when applied in this space, named the Trajectory Parameter Space (TP-Space). We also formalize the transformation between this space and the robot physical space introducing the Parameterized Trajectory Generator (PTG), a translation between both spaces by means of a family of parameterized trajectories. This formalization is addressed in a generalized form to allow us deriving any number of different transformations. Unlike previous non-holonomic approaches that use just one single transformation, the proposed method considers a variety of them simultaneously which becomes an obvious improvement to reactive approaches: each one can detect a collision-free path that the others can not. We present some experimental results to show the suitability of our method and its advantages compared with traditional approaches.**

Index Terms* – *Mobile robots, reactive navigation, non-holonomic kinematics, motion planning.

I. Introduction

For mobile robots to navigate securely and dependably, it is essential to use some geometrical space model where sensed obstacles could be represented easily and collision free paths found efficiently. In particular, reactive navigation methods can be applied on these spaces to steer robots towards target positions while avoiding collisions, based solely on sensed data and without a previous model of the environment. The space that most completely describes a robot is the Configuration Space (C-Space), which considers all the robot degrees of freedom and has been extensively used in many fields including robotics manipulators [8], maneuvers planning [7], and motion planning [11]. Nevertheless, its high dimensionality makes difficult its direct application to navigation into unknown or dynamic environments where real-time requirements demand more efficient solutions, i.e. reactive navigation methods. The majority of methods dealing with non-holonomic robots assume a family of circular paths (which are compatible with most wheel-driven and non-holonomic robots) from which to select the robot movement at each instant of time ([6],[13]). Recently, the Ego-Kinematic Space (EK-Space) [10] has been proposed to simplify the C-Space by means of the same circular paths (arcs) but in a more elegant and mathematically sound way: C-Space is reduced to EK-Space, a 2D parameter space which parameterizes the family of circular paths. Straight lines in the resultant space represent arcs to the actual robot, thus the robot can be dealt as a "free-flying-point" (without non-holonomic constraints). TP-Space can be seen as a slightly different parameter space than EK-Space. However, the restrictive use of a fixed circular paths model in [10] limits the exploration of the free-space.

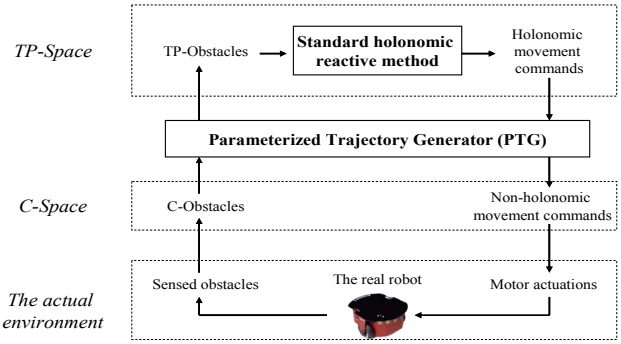

Fig. 1 The simplest application of TP-Space to reactive navigation involves using a PTG to translate obstacles into TP-Obstacles. After that, a holonomic reactive navigation method is executed in that space, and the resulting motion is then translated back into a feasible movement for the actual robot.

In this work we are motivated by the intuitive idea that, by means of removing this restriction and considering other paths models apart of the circular one, more valuable information about the free-space can be provided to the reactive method. This has been confirmed with experimental results, as will be exposed later.

For our purposes we introduce in this paper a new space representation, the Trajectory Parameter Space (TP-Space), which reduces the dimensionality of the navigation problem through a parameterization of C-Space paths (three-dimensional points) using only two parameters. Instead of considering only circular paths as previous methods, our

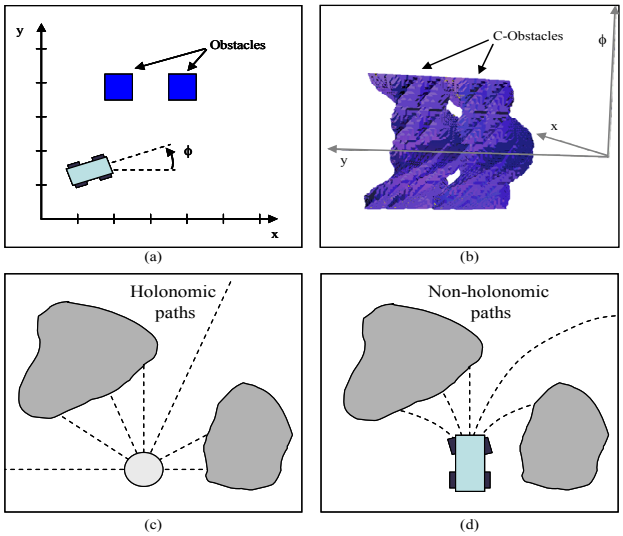

Fig. 2 In (a)-(b) two obstacles are represented in the WS (2D) and in C-Space (3D), respectively. To transform the navigation problem back into 2D, previous reactive methods use a family of paths to measure the distance to obstacles. According to the robot being holonomic (c) or not (d) classical path models are straight lines and circular arcs, respectively.

approach can manage a variety of path models, providing a more effective detection of free-space.

As potential path models we consider parametrically generated families, a concept that we formalize with the introduction of a Parameterized Trajectory Generator (PTG): a mapping between the TP-Space and a family of paths in the C-Space[1]. Our generalized formulation overcomes the circular paths limitation, thus allowing us to detect collision-free paths unavailable to previous methods. Furthermore, many different paths model can be simultaneously used to detect good paths, a feature that can not be found in previous reactive methods. In fact, it should be notice that the insights behind TP-Space can also be applied to reactive approaches to high-dimensional problems (e.g. manipulators), although in this work we focus on its application to mobile robots planar navigation only.

To clarify the roles of TP-Space and PTG a simplified TP-Space based reactive navigation system is sketched in Fig. 1. The main steps involved are:

1. The robot senses obstacles, which in the robot C-Space generate C-Obstacles.
2. A given PTG translates obstacles into TP-Space (TP-Obstacles are obtained).
3. A holonomic reactive navigation method selects a desired motion in TP-Space, assuming a robot without non-holonomic constraints.
4. The PTG translates the holonomic motion into a non-holonomic one, feasible for the actual robot.
5. Repeat from step 1 until target location is reached.

In practical systems a variety of PTGs can be used simultaneously and a selection mechanism must be introduced to dynamically choose the most suitable one at each instant of time (this is not addressed in the present paper).

The above steps are explained throughout the paper, which is organized as follows. In section II we review other representation spaces, and TP-Space is introduced in section III. Next we describe our formulation of a PTG and how it is applied for building TP-Obstacles and extracting non-holonomic motion commands from TP-Space. Experimental results from real robots are presented in section V. We finalize with some conclusions.

II. Other Space Representations

The Workspace (WS) is the simplest representation space. It contains the set of reachable locations in the robot environment. For most wheeled robots the WS is commonly assumed to be a 2D plane, thus the robot heading is ignored. As a result, WS can be applied precisely only to circular shaped robots, certainly a too restrictive condition for many practical situations. However, sensed obstacles can be easily represented into WS since they are usually modelled as 2D obstacle-points ([1],[4],[9],[13]).

A more complete robot description is achieved with C-Space ([7],[8]) whose complexity depends on the number of degrees of freedom. For our purposes the robot can be represented by its position (x,y) and its orientation ϕ, obtaining a three-dimensional C-Space. In this space the robot is represented as a single point and obstacles (C-Obstacles) implicitly hold the robot shape, as illustrated in Fig. 2(a)-(b). Robot navigation therefore becomes a path-finding problem in this three-dimensional space. The construction of the whole C-Obstacles is a demanding computational process since all the poses that make the robot to collide with any obstacle must be found. On the other hand, robots moving in dynamic and cluttered environments must respond quickly to environment changes, making impractical the direct application of C-Space in these situations. This is the motivation for reactive navigation methods, which can be mainly classified into:

- Holonomic methods, working directly in WS and computing a desired motion based on currently sensed obstacles, as the ones reported in [2],[4],[14], and [9].
- Non-holonomic methods, most of them working in the velocity space ([1],[12],[13]) and some of them also including the dynamic window concept [5].

These approaches have different limitations. Holonomic ones are not directly applicable to most existing car-like robots, while non-holonomic methods based on the dynamic window ignore part of the sensed information. Velocity space methods by their part use circular paths only to estimate distance to obstacles which limits the possibility to detect a better motion for the robot. Actually, a key operation in all reactive approaches is that of estimating the distance until collision. Next, we provide some insight into this issue since its generalization makes possible the introduction of TP-Space.

[1] We use the term *trajectory* instead of *path* since our approach considers the time and the robot velocities, as exposed later.

III. THE TP-SPACE

A. Problem statement

Reactive methods decide movements based on current sensed data only, from which distance until collision is estimated along a family of paths. This operation becomes a central issue in reactive navigation since it provides the only source of information about the free-space (see Fig. 2). For holonomic robots, which can move in any direction, these distances are measured over straight paths, while for non-holonomic robots, which move along a sequence of infinitesimal circular arcs, distances are measured along circular paths (see Fig. 2(c)-(d)). Although the plenty of previous methods consider only this pair of path models, they are just two ways out of the infinity of them that could be employed for this purpose. In the following we provide some insight into the obstacles sampling mechanism and the importance of path models and distance measurement in this process.

We define a robot path as a continuous sequence of locations and orientations, thus it can be plotted as a three-dimensional curve in C-Space (as the one in Fig. 3). A family of parametrically generated paths in this space will hence generate a surface, as the example in Fig. 4. The intersection of this *sampling surface* with the C-Obstacles indicates the poses of the robot that lead to collisions. In other words, the obstacles sampling should be considered in C-Space as using a different 3D shape for each path model. Rigorously, distances to collision should be measured along the respective C-Space paths. This can not be determined with the usual Euclidean metric since useless values are obtained: dimensions of different nature, linear and angular, are added together, as sketched in Fig. 3. Previous approaches avoid this problem by measuring Euclidean distances along the paths projection on WS but ignoring changes in the robot orientation. To be rigorous, we propose in the next section to replace the Euclidean metric in C-Space with a more appropriate metric.

B. Definition of TP-Space

TP-Space is a two-dimensional space where a point, given by its polar coordinates (α,d), corresponds to another one on a sampling surface in C-Space. More precisely, α defines a trajectory out of a given family, and d represents the distance along that trajectory. In practice, we are only interested in the circle of unit radius, that is, in the range $A\times D\subset\mathbb{R}^2$, where $A=\{\alpha \mid \alpha \in]-\pi,\pi]\}$ and $D=\{ d \mid d \in [0,1]\}$, provided that distances are normalized with respect to an arbitrarily large distance d_{MAX}.

The purpose of using TP-Space as a space representation is to convert the free-space sampling problem of non-holonomic robots (in C-Space) to one for a holonomic robot in the TP-Space, which can be considered a virtual Workspace (WS). Since this transformation is applied at each iteration, the robot will always be at the origin and surrounding obstacles will appear within the unit circle. In this virtual WS, well-known holonomic reactive methods for circular robots

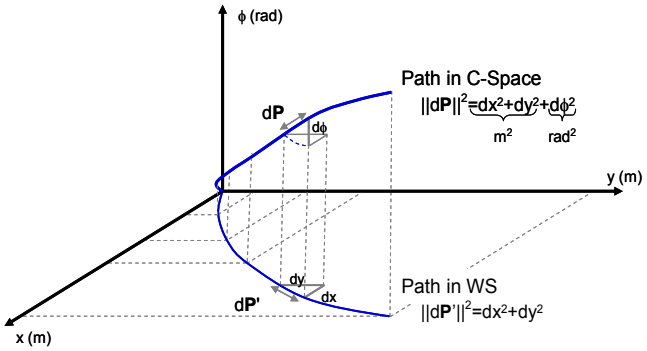

Fig. 3 The metric space for WS is well defined by taking Euclidean distance as its metric, but this is not acceptable in C-Space due to the orientation dimension. This forces the introduction of a more appropriate metric.

can be applied. These methods are the simplest and most efficient reactive navigation approaches, thus, the benefit of being able to apply them to any kind of robot (non-circular or/and non-holonomic) is clear. As shown in Fig. 4, a straight path given by a constant α value in TP-Space is equivalent to a trajectory of a certain family (or sampling surface), which is defined by a given PTG.

IV. THE PARAMETERIZED TRAJECTORY GENERATOR (PTG)

A. Definitions

A PTG is a mapping from TP-Space points (α,d) to C-Space poses $((x,y),\phi)$, such that straight paths from the origin in TP-Space are transformed into kinematics-compliant paths in C-Space (see Fig. 4). Formally, a PTG is defined as:

$$\text{PTG:}\quad A\times D \subset \mathbb{R}^2 \rightarrow \mathbb{R}^2\times A \qquad (1)$$

$$(\alpha,d) \rightarrow \{(x,y),\phi\}$$

where $A=\{\alpha|\alpha\in]-\pi,\pi]\}$ and $D=\{d|d\in[0,1]\}$ define the domain of the PTG. In order to generalize as much as possible the generation of paths, we propose to describe them in terms of trajectories. A trajectory for a robot is just like a path but taking time into account, which is required for linear and angular velocities to be defined at each point. The time used in a PTG may not be the robot actual time since velocities can be scaled for speed control purposes, as commented later on. Actually, time (t) is introduced as a substitute of the distance (d) in the PTG. There are two reasons for this change of variable: (i) for having a physical meaning: the natural parameterization of trajectories is time; and (ii) C-Space lacks of a proper metric for distances, in the sense that was discussed in the previous section. The change of variable therefore involves the definition of a custom non-Euclidean metric in such a way that a one-to-one correspondence between distance and time is established. Before exposing the details of this custom metric, the C-Space variables defined by a PTG must be introduced. Let $\mathbf{V}(\alpha,t)=[v(\alpha,t)\ \omega(\alpha,t)]^T$ be the velocity vector over a given trajectory in TP-Space, where the components are the linear and angular velocities of the robot, respectively. The definition of the functions $v(\alpha,t)$ and $\omega(\alpha,t)$, denominated *design functions* of the PTG, will state the mapping between the TP-Space and the C-Space, since its

integration over time yields the robot trajectory (in C-Space). More concretely, let $x(\alpha,t)$, $y(\alpha,t)$ and $\phi(\alpha,t)$ be the components of the robot pose (configuration), defined as $\mathbf{P}(\alpha,t)=[x(\alpha,t)\ y(\alpha,t)\ \phi(\alpha,t)]^T$. These poses are then obtained by integration of the kinematics constraint equation, which for the case of a car-like robot stands as,

$$\frac{\partial \mathbf{P}(\alpha,t)}{\partial t} = \begin{bmatrix} \cos\phi(\alpha,t) & 0 \\ \sin\phi(\alpha,t) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(\alpha,t) \\ \omega(\alpha,t) \end{bmatrix} \qquad (2)$$

where the right-most vector is the *velocity vector* $\mathbf{V}(\alpha,t)$ whose design will totally determine the properties of the resulting PTG. The expression (2) is not integrable, hence an analytical expression for $\mathbf{P}(\alpha,t)$ can not be supplied in the general case and numerical solutions must be used[2].

The problem of supplying a proper metric for C-Space can now be addressed. We propose the following metric for a pose increment $\Delta\mathbf{p}=[\Delta x\ \Delta y\ \Delta\phi]^T$ in C-Space:

$$m(\Delta\mathbf{p}) = \left\| \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & r \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\phi \end{bmatrix} \right\| = \sqrt{\Delta x^2 + \Delta y^2 + (r\cdot\Delta\phi)^2} \qquad (3)$$

where the change in the robot orientation is incorporated into the measurement by means of a given distance r (this constant parameter can be set to any value in the order of the robot size). Applying this metric to a given trajectory means to measure the traveled distance from the origin to any given point on that trajectory, taking also into account the normalization distance (d_{MAX}), since we are interested in normalized distances in the range [0,1]. Let this distance measurement be denoted as $\mu_\alpha(t)$ for the trajectory α and until time t. An expression for this trajectories metric can be obtained from:

$$\begin{aligned} \mu_\alpha(t) &= {}^1\!/_{d_{MAX}} \int_0^t m\left(\frac{d\mathbf{P}}{d\tau}\right) d\tau = \\ {}^1\!/_{d_{MAX}} &\int_0^t \left\| \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & r \end{bmatrix} \begin{bmatrix} \cos\phi(\alpha,t) & 0 \\ \sin\phi(\alpha,t) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(\alpha,t) \\ \omega(\alpha,t) \end{bmatrix} \right\| d\tau = \\ {}^1\!/_{d_{MAX}} &\int_0^t \sqrt{v(\alpha,\tau)^2 + (r\cdot\omega(\alpha,\tau))^2}\, d\tau \end{aligned} \qquad (4)$$

where the metric proposed in (3) has been used instead of the Euclidean norm. This function establishes a bijective mapping between time t and the normalized distance d in a PTG. Hence the change of variable is solved and the general expression for a PTG can finally be written down as:

$$PTG(\alpha,d) = \mathbf{P}\left(\alpha, \mu_\alpha^{-1}(d)\right) \qquad (5)$$

where poses $P(\alpha,t)$ are solved numerically for each PTG. The inverse process, required for constructing the TP-Obstacles, translates C-Space poses into TP-Space and is given by the inverse *PTG* function:

[2] A prominent exception is the case where velocities are constant, that is, when the robot moves along circular paths.

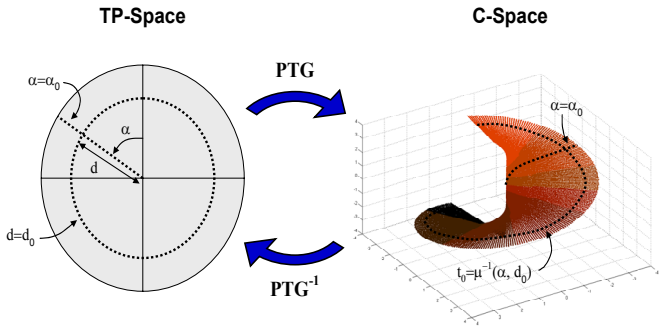

Fig. 4 When a family of paths in C-Space is generated parametrically using a given PTG, a characteristic 3D shape appears. A PTG is designed for transforming radial paths in TP-Space into feasible paths in C-Space.

$$PTG^{-1}: Surf(\text{PTG}) \subset \mathbb{R}^2\times A \rightarrow A\times D \qquad (6)$$
$$((x,y),\phi) \rightarrow (\alpha,d)$$

whose domain is not the whole C-Space but only *Surf*(PTG), that is, the points of the 3D sampling surface:

$$Surf(\text{PTG}) = \{\mathbf{P} \mid \mathbf{P}=PTG(\alpha,d)\}\ \ \forall\ (\alpha,d)\in A\times D \qquad (7)$$

B. Building TP-Obstacles

A TP-Space based navigation system performs two transformations with each PTG: (i) the construction of TP-Obstacles, and (ii) the translation of the target location into TP-Space. Both transformations imply mapping WS points into TP-Space but they represent quite different processes. The translated target location remains being a single point in TP-Space and it is simply computed through the $\Gamma(\cdot)$ function, defined as:

$$\Gamma(x_0,y_0) = \arg\min_{(\alpha,d)} \left\| \begin{bmatrix} x_0 \\ y_0 \\ 0 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot PTG(\alpha,d) \right\| \qquad (8)$$

and finding the closest trajectory to any given location (ignoring the robot orientation).

On the other hand, obstacle points are translated into TP-Space regions, namely TP-Obstacles. Let $\mathbf{o}\in\mathbb{R}^2$ be a real obstacle point in WS, and *C-Obstacle*($\mathbf{o}$) its representation in C-Space. We define the *TP-Obstacle* for the point $\mathbf{o}$ as:

$$TP\text{-}Obstacle(\mathbf{o}) = \{\ (\alpha,d) \mid (\alpha,d)=PTG^{-1}(\mathbf{P}), \qquad (9)$$
$$\forall\ \mathbf{P}\in C\text{-}Obstacle(\mathbf{o}) \cap Surf(PTG)\ \}$$

that is, it is defined as the translation of the intersection between C-Obstacles and the sampling surface of the PTG into TP-Space. Since a robot can collide with obstacles, even those ones consisting of a single point, from many different poses, *TP-Obstacles* are always two-dimensional regions in TP-Space, even when generated from a single obstacle point.

Since holonomic methods in TP-Space will measure obstacle distances along radial paths we are only interested in the closer obstacle in each direction, thus only the minimum collision-free distance for each α value must be kept (recall that constant α values results in straight paths in TP-Space).

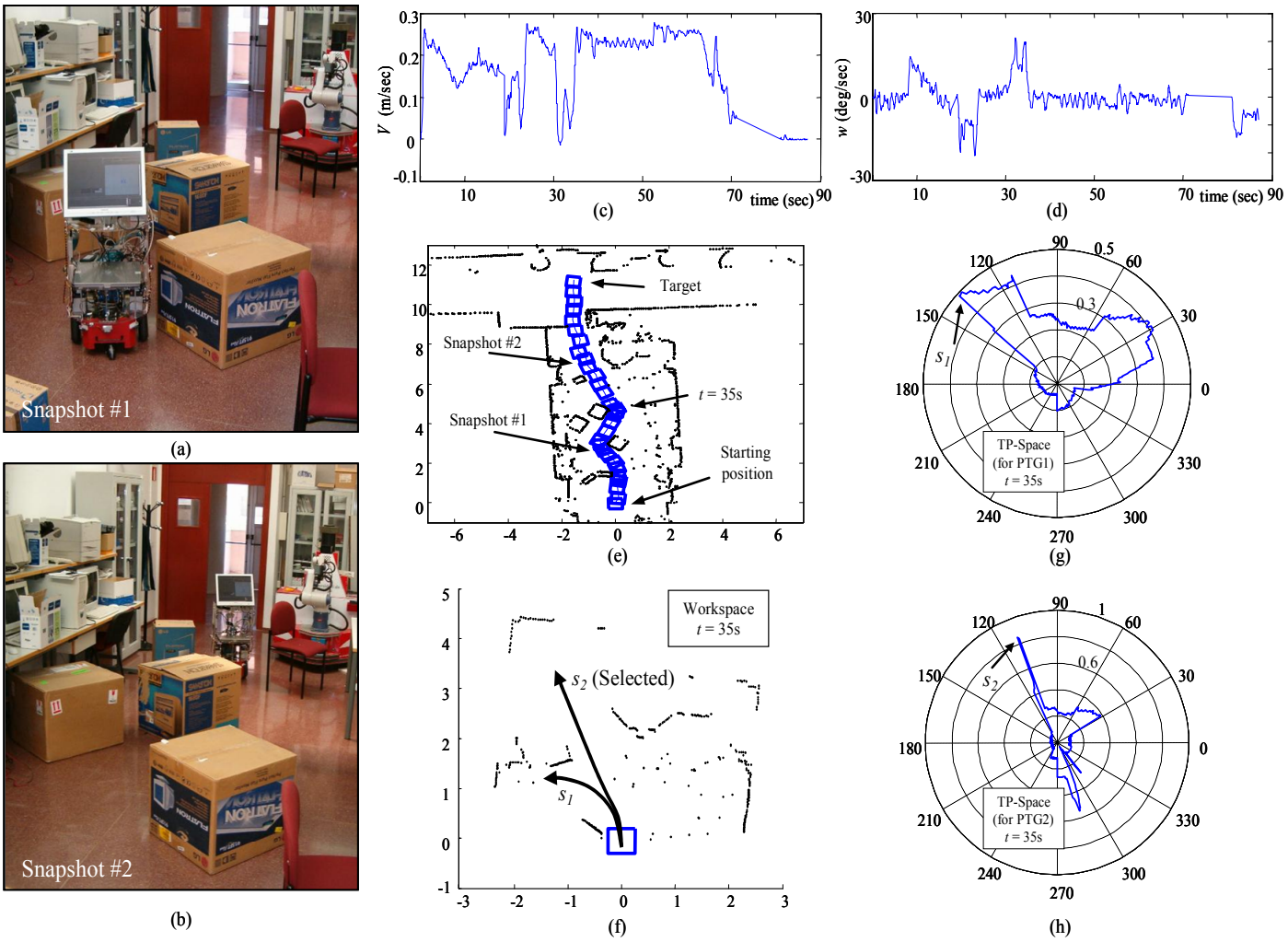

Fig. 5 (a)-(b) Snapshots from the navigation experiment using our reactive method, which generates the linear and angular velocity commands that are plotted in (c) and (d), respectively. The path followed by the robot towards the target can be seen in (e), where also the instants of time corresponding to the snapshots are highlighted. For another instant of the experiment, t=35sec, the sensed obstacles are shown in (f) in the WS, whereas their translations into TP-Space are in (g)-(h) for a classical paths-based PTG and a custom one, respectively. As further explained in the text, the non-circular PTG provides a better movement here.

C. From TP-Space movements to real ones

The motion command generated by a holonomic method in the virtual WS (TP-Space) is a pair stating the robot desired speed s and direction α_m (in the interval]-π,π]). This motion command must be translated back to a non-holonomic movement, which can be carried out in two steps:

1. We obtain a normalized velocity command from the evaluation of the PTG design function at $\alpha=\alpha_m$, that is:

$$\mathbf{V}_{\text{norm}}(\alpha_m) = \mathbf{V}(\alpha_m,0) = [v(\alpha_m,0)\ \omega(\alpha_m,0)]^{\text{T}} \qquad (10)$$

   We take the initial response (at t=0) since the PTG reference system is the robot current pose, i.e. the robot is always at the origin of trajectories in the TP-Space.
2. The non-holonomic command velocity $\mathbf{V}_{\text{n-h}}$ (in the real world) is obtained by scaling $\mathbf{V}_{\text{norm}}$ according to the holonomic velocity s in TP-Space (provided by the holonomic method). Mathematically:

$$\mathbf{V}_{n-h}(\alpha_m, s) = \frac{s}{\max_{\alpha}\{m(\mathbf{V}(\alpha,0))\}} \cdot \mathbf{V}_{norm}(\alpha_m) \qquad (11)$$

where $m(\cdot)$ is a custom metric for speeds similar to that defined in (3) for pose increments:

$$m\left(\begin{bmatrix} v \\ w \end{bmatrix}\right) = \left\| \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & r \end{bmatrix} \begin{bmatrix} \cos\phi & 0 \\ \sin\phi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \right\| = \sqrt{v^2 + (r\cdot\omega)^2} \qquad (12)$$

D. Practical implementation issues

Two main problems must be addressed to get a functional TP-Space based reactive navigation system: (i) obtaining a solution to equations for each PTG; and (ii) building TP-Obstacles. The first problem consists on solving (2), which is easily accomplished with standard numerical techniques for a discrete set of a values. Building the TP-Obstacles is a more complex process, but a fast method based on lookup tables has been developed following ideas related to those in [12]: the WS is arranged as a rectangular grid and each cell stores the

TP-Obstacle for obstacles into that cell. As a result, translating the real obstacles becomes just adding the elemental TP-Obstacles for the occupied cells. Then, just the closer obstacle for each α value must be kept, which will be passed to the holonomic reactive method as obstacle distances in the virtual WS. The required time for building TP-Obstacles with this technique is linear in the number of points. In spite of that building the tables takes a long time (almost a minute), they must not be updated as long as neither the kinematics nor the shape of the robot change.

Concerning the holonomic method to be applied in TP-Space in our implementation we use a custom design inspired in [9]. Our navigation system considers a variety of PTGs, and consequently it includes a method for choosing the most promising transformation based on a weighted average of a number of objective functions which are evaluated for each possibility. We do not provide here more details of neither the selection nor the holonomic navigation methods because of the space limitation.

V. Results and Conclusions

TP-Space based reactive navigation has been tested for almost a year of extensively use in different scenarios including corridors, cluttered rooms, and open spaces. In our implementation we currently use two different PTGs, one of them generating circular paths and the other based on heuristic rules (the 3D sampling surface for this PTG is plotted in Fig. 4). This approach has been tested on two different robotic platforms: SENA, a robotic wheelchair [5], and *Sancho*, a service robot built on a Pioneer 3 mobile base. The whole navigation system is the same for both robots except for the robot shape and the speeds limits, which are used in building the lookup tables. Fig. 5 shows the record of a navigation using *Sancho*, where the reactive navigation system is commanded for moving towards a point located outside the room. The twisty trajectory followed by the robot is shown in Fig. 5(e) over a map where obstacles are also drawn. It can be appreciated that the robot gets out the room smoothly, which shows that even the narrow free-space through the door is clearly visible in TP-Space. In order to demonstrate the improvement that our approach brings into the navigation, let's take a look at a given point of the navigation, concretely, the one pointed out in Fig. 5(e) for t=35s. At this point, the sensed obstacles are the ones plotted in Fig. 5(f), and TP-Obstacles corresponding to the circular paths and to the custom PTG are shown in Fig. 5(g)-(h), respectively. The holonomic reactive method selects s_1 and s_2 as the best straight paths in each one of these virtual-WS (TP-Space), whose translations into actual trajectories are highlighted in Fig. 5(f). It is manifest that traditional approaches (implicitly using our circular PTG) can not find the collision-free path shown in the latter, which is clearly more advantageous. Although any path can be decomposed as a sequence of arcs, our method provides directly the reactive algorithm with valid collision-free trajectories that may not been detected with circular arcs only.

Regarding the time efficiency of our method, the TP-Obstacles construction is the most time demanding stage in the whole process. However, its running time turns out to be quite modest: it takes 0.87ms for the on-board computer (Pentium M, 1.8GHz) to translate the 361 obstacle points that are supplied by a 2D laser range scanner. The grid size is 7x7 meters, cells into this grid are 2x2cm (122,500 cells), and 512 discrete values for α are taken into account for the lookup-tables building. With these settings, our robots are able to detect openings as small as 4 cm larger than the robot size.

Finally, we outline the following conclusions. We have presented a new approach to the non-holonomic reactive navigation problem that allows a clear and useful separation of the usually merged problems of selecting non-holonomic compliant motions and choosing it to avoid collisions. This is accomplished through a space transformation into TP-Space, whose implementation has shown to be effective and efficient. Unlike other transformed spaces, the transformation is defined in the general form of a PTG, allowing any number of custom transformations, each one carrying a portion of information about C-Obstacles, while the simultaneously use of a variety of PTGs increases the effective use of sensed data. Further research is needed in order to provide custom PTG designs and to integrate the robot dynamics into the approach.

References

[1] K.O. Arras, J. Persson, N. Tomatis, R. Siegwart, "Real-Time Obstacle Avoidance For polygonal Robots With a Reduced Dynamic Window," in *Int. Conf. on Robotics and Automation*, 2002.

[2] J.L. Blanco, J. Gonzalez, J.A. Fernández-Madrigal, "The TP-Space: Foundations and applications," *Technical Report*, Dept. of System Engineering and Automation, University of Malaga, 2005.

[3] J. Borenstein, Y. Koren, "Real-time Obstacle Avoidance for Fast Mobile Robots," in *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no.5, pp.1179-1187, 1989,.

[4] J. Borenstein, Y. Koren, "The Vector Field Histogram – Fast Obstacle Avoidance for Mobile Robots," in *IEEE Transactions on Robotics and Automation,* 1991.

[5] J.A. Fernández-Madrigal, C. Galindo, J. González, "Assistive Navigation of a Robotic Wheelchair using a Multihierarchical Model of the Environment," in *Integrated Computer-Aided Engineering*, v.11, no.4, pp.309-322, 2004.

[6] D. Fox, W. Burgard, S. Thrun, "The dynamic window approach to collision avoidance," in *IEEE Robotics and Automation Magazine*, 1997.

[7] C. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, 1991.

[8] T. Lozano-Pérez, "A Simple Motion-Planning Algorithm for General Robot Manipulators," in *IEEE Journal of Robotics and Automation*, 1987.

[9] J. Minguez, L. Montano, "Nearness Diagram (ND) Navigation: Collision Avoidance in Troublesome Scenarios," in *IEEE Transactions on Robotics and Automation*, v. 20, no. 1. Feb, 2004.

[10] J. Minguez, L. Montano, and J. Santos-Victor, "Reactive navigation for non-holonomic robots using the ego-kinematic space," in *Int. Conf. on Robotics and Automation*, v.3, pp. 3074–3080, 2002.

[11] R.R. Murphy, *Introduction to AI Robotic*, the MIT Press, 2000.

[12] C. Schlegel, "Fast Local Obstacle Avoidance under Kinematics and Dynamic Constraints for a Mobile Robot," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 1998.

[13] R. Simmons, "The Curvature-Velocity Method for Local Obstacle Avoidance," in *Int. Conf. on Robotics and Automation*, 1996.

[14] P.K. Pal, A. Kar, "Mobile Robot Navigation Using a Neural Net," in *Int. Conf. on Robotics and Automation*, 1995.