

Extending Obstacle Avoidance Methods through Multiple Parameter-Space Transformations*

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

Abstract

Obstacle avoidance methods approach the problem of mobile robot autonomous navigation by steering the robot in real-time according to the most recent sensor readings, being suitable to dynamic or unknown environments. However, real-time performance is commonly gained by ignoring the robot shape and some or all of its kinematic restrictions which may lead to poor navigation performance in many practical situations.

In this paper we propose a framework where a kinematically constrained and any-shape robot is transformed in real-time into a free-flying point in a new space where well-known obstacle avoidance methods are applicable. Our contribution with this framework is twofold: the definition of generalized space transformations that cover most of the existing transformational approaches, and a reactive navigation system where multiple transformations can be applied concurrently in order to optimize robot motion decisions. As a result, these transformations allow existing obstacle avoidance methods to perform better detection of the surrounding free-space, through “sampling” the space with paths compatible with the robot kinematics.

We illustrate how to design these space transformations with some examples from our experience with real robots navigating in indoor, cluttered, and dynamic scenarios. Also, we provide experimental results that demonstrate the advantages of our approach over previous methods when facing similar situations.

1 Introduction

Autonomous and safe navigation is undoubtedly one of the issues that need to be rigorously solved for mobile robots to leave research labs and generalize their use. For this purpose, it is essential to account for some spatial representation of obstacles where collision-free paths could be found efficiently. This problem has been extensively studied by the robotics community and has traditionally led to two different research areas. On the one hand we have *motion planning* approaches, where an optimal path from a start point to a target is computed for a known scenario. The Configuration Space (C-Space) (Lozano-Pérez 1987) has been successfully employed as representation in this scope. In C-Space the robot is represented as a single point at the cost of high-dimensionality (for example, three dimensions for a planar mobile robot). On the other hand, some navigation approaches deal with unknown or dynamic scenarios, where motion commands must be periodically computed in real-time during navigation (there is no planning).

* A shorter version of this article appeared in the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2006.

Under these approaches, called *reactive* or *obstacle avoidance*, the navigator procedure can be conveniently seen as a mapping between sensor readings and motor actuations (Arkin 1998). Although reactive methods are quite efficient and have simple implementations, many of them do not work properly in practical applications since they often rely on too restrictive assumptions, like a point or circular representation of the robot or allowing movements in any direction, i.e. ignoring kinematic restrictions. C-Space is not an appropriate space representation for reactive methods due to its complexity, which prohibits real-time execution. Hence simplifications of C-Space have been proposed specifically for reactive methods. Finally, combinations of the two above approaches have also been proposed (Khatib et al. 1997, Lamiraux et al. 2004, Quinlan and Khatib 1993), which usually start computing a planned path based on a known static map, and then try to dynamically deform it to avoid collision with unexpected obstacles. These hybrid approaches successfully solve the navigation problem in many situations, but purely reactive methods are still required for partially known or highly dynamic scenarios, where an a priori planned path may need excessive deformation to be successfully constructed by a hybrid method.

This paper presents a *purely reactive* approach to the navigation problem, thus previous works on motion planning and paths deformation are not directly related to ours. More concretely, the problem we address here is the reactive navigation of kinematically-constrained, any-shape mobile robots in planar scenarios. This problem requires finding movements that approach the target location while avoiding obstacles and fulfilling the robot kinematic restrictions.

Our main contribution lies on the process for detecting free-space around the robot, which is the basis for a reactive navigator to decide the best instantaneous motor actuation. For this task, existing methods consider certain families of simple paths for measuring obstacle distances (which is equivalent to sampling the free-space). These families of paths, that we will call *path models*, must be considered not as planned paths but as artifacts for taking nearby obstacles into account. All existent reactive methods use path models that are an extension of the robot short-term motor actuations, as illustrated in Fig. 1: for holonomic robots, which can move in any direction, straight paths are employed, whereas circular arcs are considered for non-holonomic ones. Apart from straight and circular path models used in previous reactive methods, other models have been studied in the field of motion planning, and have obtained interesting results

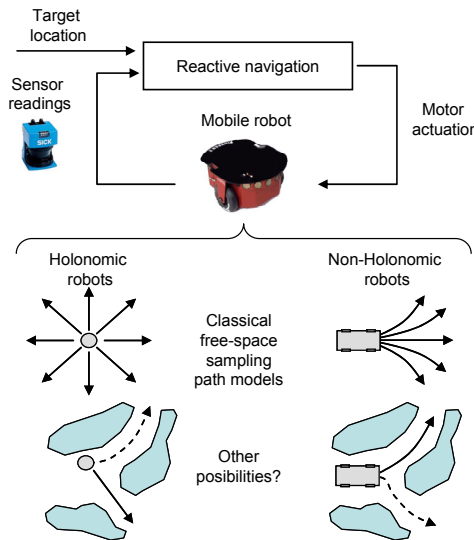


Fig. 1 Reactive navigation methods periodically map sensor readings to motor actuations in order to avoid obstacles and to reach a target location. For taking obstacles into account, straight and circular sampling path models have been used for holonomic and non-holonomic robots, respectively. In this paper we claim that other valid possibilities exist (dashed curves are examples) which provide the robot with a more comprehensive free-space detection.

regarding shortest paths (Laumond and Souères 1993, Reeds J.A. and Schepp R.A. 1990, Souères and Laumond 1996, Vendittelli et al. 1999). Our approach allows some of those methods to be integrated into a reactive navigation system for the first time.

We claim that straight and circular paths, used in previous reactive methods, are just two out from the infinity of path models that can be followed by a robot in a memoryless system, that is, reactively: It is worth to mention that a path model used in a reactive navigator can not be an arbitrary one since it must both satisfy the kinematic constraint and be able of being described by a reactive controller. It is clear that considering other path models is more appropriate to sample the free-space than using the classic straight or circular models only. We shed light on this issue through the example in Fig. 2, where a robot (reactively) looks for possible movements. If we employ a single circular path model for sampling obstacles as in Fig. 2(a), it is very likely that the obstacle avoidance method overlooks many good potential movements – notice that any reactive method must decide according solely to the information that path models provide about obstacles. In contrast, using a diversity of path models, as the one shown in Fig. 2(b), provides the robot a more complete view of the free-space. This is one of the distinctive features of our approach: the capability to handle a variety of path models simultaneously. As discussed later on with real experiments, this reflects into the robot carrying out navigation in less time following shorter paths. Additionally, the navigation performs more robustly against dynamic obstacles and cluttered scenarios.

The problems of kinematic restrictions and obstacle avoidance can be decoupled by using path models to transform kinematic-compliant paths and real-world obstacles into a lower complexity space, which we will name Trajectory Parameter Space (TP-Space for short)*. This transformation is defined in such a way that the robot can be considered as a free-flying-point in the TP-Space since its shape and kinematic restrictions are already embedded into the transformation process. We can then entrust the obstacle avoidance task in the transformed space to any standard method for free-flying-point robots. We have found this idea in (Minguez and Montano 2006) where the Ego-Kinematics Transformation (EKT) is presented as a concrete TP-Space. In the context of that work, our contribution here is two-fold: on the one hand, we extend their work with the generalization of path models through a novel tool called Parameterized Trajectory Generator (PTG), which permits us to handle any number of transformations. On the other hand, we propose a navigation system to manage simultaneously multiple paths, each corresponding to a different PTG. This system faces the dynamic selection of the best alternative at each instant of time (for instance, in terms of path length and clearance), which yields a more

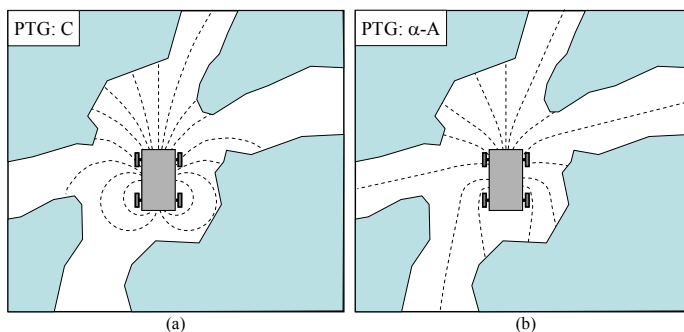


Fig. 2 A motivating example for our approach. While previous reactive methods for non-holonomic robots consider circular paths only, as in (a), we propose to account for other possible path models that may be able to find better collision-free paths, for example, the α -A model shown in (b), which will be described in this paper.

* Notice the use of the term *trajectory* instead of *path*. As exposed below, this is because we will deal with both robot poses and velocities.

powerful approach than traditional methods relying on circular paths only.

To sum up, the proposed solution to reactive navigation presents the following features:

- The problems of obstacle avoidance and kinematic restrictions are decoupled like in (Minguez and Montano 2006), but now in a generalized way that makes easier their optimization and reusability between different robots and scenarios.
- Well-known, efficient reactive methods previously constrained to holonomic point (or circular) robots can now be applied to any-shape, non-holonomic ones.
- Using path models other than circular arcs enables the robot to find a wider set of alternative motions that avoid obstacles and are compatible with kinematic restrictions. Furthermore, a number of path models can be used simultaneously and the one that best suits to each specific situation can be chosen dynamically.

In Fig. 3 we represent schematically both a classical reactive navigation method and our approach. Existing methods deal with the kinematic restrictions of a robot and obstacle avoidance as a whole, as illustrated in Fig. 3(a). We propose to abstract the kinematic restrictions from the collision avoidance through a number of different transformations (PTGs). As shown in Fig. 3(b), the existence of multiple transformations introduces the need for some sort of selection mechanism for the most appropriate movement at each time step.

The rest of this article is organized as follows. In section II we review different space representations employed in previous approaches to deal with the problem of reactive navigation with kinematic restrictions. In section III we introduce TP-Spaces and the underlying distance-to-obstacles measuring problem. Next we define PTGs and how to make use of them for the needed transformations between the real navigation scenario and a TP-Space. In section V the complete navigation system is presented, including some implementation details from our experience with real robots. Finally, some results and conclusions are discussed.

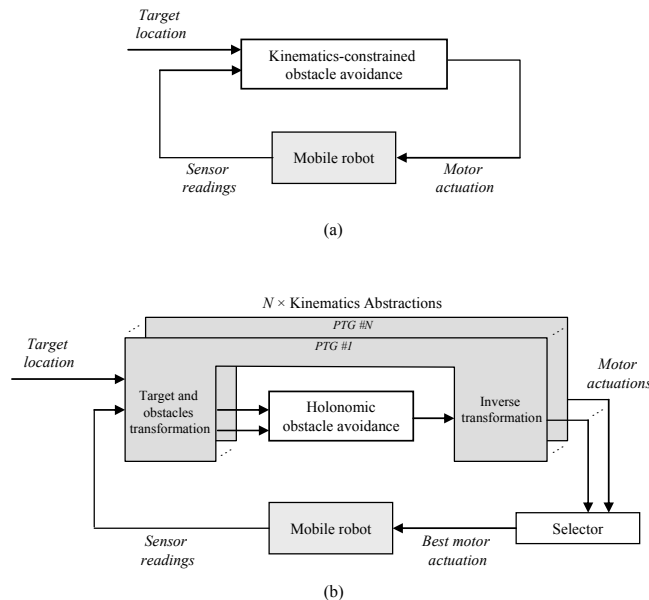


Fig. 3 Instead of considering the robot kinematics into the reactive method itself like most existing approaches do (outlined in (a)), we propose here to use a number of different kinematics abstractions layers, as shown in (b). A simple holonomic method is executed for each one of the transformed scenarios.

2 Relation with previous works

In the following we review the most well-known space representations that have been employed in mobile robot motion planning and collision avoidance.

The C-Space has been extensively used in many fields, including robotic manipulators (Lozano-Pérez 1987), maneuver planning (Latombe 1991), and mobile robot motion planning (Murphy 2000). In spite of recent advances towards speeding up its computation (Zhang et al. 2006), the complexity derived from its high dimensionality makes C-Space not applicable to real-time reactive navigation. The problem can be visualized with the example of Fig. 4(a), where it is shown a path for a robot in a planar scenario with an obstacle. This situation can be translated into C-Space as an obstacle that implicitly holds the robot shape (C-Obstacle). Then the navigation problem becomes that of finding a path in this space for a point robot (see Fig. 4(b)). Unfortunately, building the whole C-Obstacles and finding paths in this higher dimensional space remains being a computationally expensive process.

A first simplification for dealing with C-Space more efficiently is to assume a circular robot. Thus, C-Obstacles are no longer dependent on the robot orientation and the C-Space reduces to a planar space that coincides with the robot workspace (WS). This approach is employed in most potential field methods, like the VFF (Borenstein and Koren 1989), VFH (Borenstein and Koren 1991), and others (Haddad et al. 1998, Balch and Arkin 1993). Other reported methodologies are based on neural nets (Pal and Kar 1995) and, more recently, the Nearness-Diagram (ND) approach (Minguez and Montano 2004), which relies on a divide-and-conquer strategy that defines a set of different states according to the arrangement of nearby obstacles. All these methods deal with circular robots, a too restrictive assumption for many real-life situations. For example, if a robotic wheelchair were assumed to be circular it would never pass through a narrow doorway.

Most approaches that deal with any-shape, kinematically-constrained robots work with another simplification of C-Space: the velocity space (Arras et al. 2002, Feiten et al. 1994, Ramirez and Zegloul 2001, Schlegel 1998, Simmons 1996), or V-Space for short. For the mobile robots of our interest, V-Space represents the space of the potential linear and angular robot velocities, hence the next movement can be chosen as a point in V-Space that results in constant curvature paths (i.e. circular arcs). A common feature in many V-Space methods is the inclusion of a dynamic window (Fox et al. 1997), which restricts the range of reachable velocities to that compatible with the robot maximum acceleration. An important limitation of these methods is that, although many obstacles may be sensed, not all of them are actually taking into account: only those ones falling into the robot dynamic window for the next step are considered for

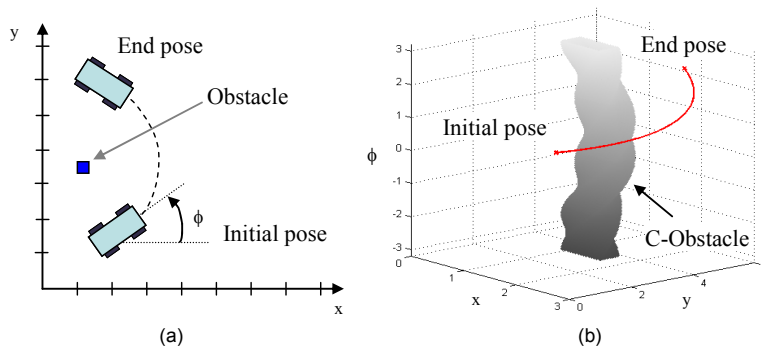


Fig. 4 (a) A robot and a path avoiding an obstacle are represented here from a top view. (b) In C-Space the robot is now represented as a point, and the path becomes a 3D curve.

choosing the instantaneous motion command. It is clear that better paths would be found if more comprehensive obstacle information were taken into account, which indeed implies looking ahead for more than one step, as our approach does.

In addition to the utilization of a dynamic window, most V-Space approaches use only the family of circular paths to sample the free-space, which entails the risk of not detecting many free-space areas. There are some exceptions (Ramirez and Zegloul 2001, Xu and Yang 2002) that make use of straight paths, but this model is not appropriate for most actual mobile robots. Only these two path models have been reported in the reactive collision avoidance literature. Interestingly, all these approaches can be seen as especial instances of our generalized space transformation approach. We must also mention the related works on velocity obstacles, introduced in (Fiorini and Shiller 1998), where a similar velocity space is used to account for the robot and obstacles movement simultaneously. Nevertheless, in this approach the trajectories are also assumed to be circular arcs and the robot and the obstacles are approximated by circles. In spite of these approximations, that approach remains being an interesting choice for motion planning in some circumstances.

An elegant and mathematically sound alternative to V-Space has been reported in (Minguez and Montano 2006) building 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 through two parameters only, namely the path curvature and the distance along the arc. In that work it is shown that, if reactive navigation is carried out in this trajectory-parameter space (TP-Space), the robot can be treated as a free-flying-point. Thus, navigation in a parameterized space allows us to decouple the problems of kinematic restrictions and obstacle avoidance. However, this approach has never been extended either to cope with other path models apart from circular arcs, or to a number of different transformations, which are the contributions of the present work.

To further clarify the relationship between the different existing representation spaces, please refer to Fig. 5, where TP-Spaces appear as a generalization of spaces such as WS and V-Space. However, it must be remarked that C-Space is the most general space representation, at the price of an elevated computational cost due to its high dimensionality.

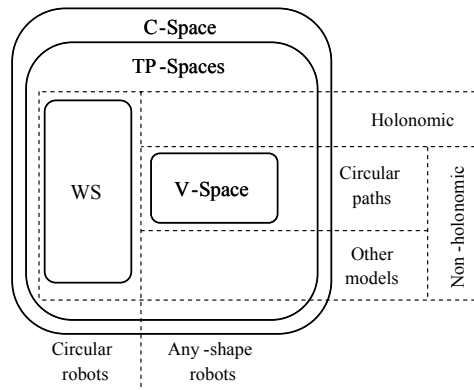


Fig. 5 A classification of different representation spaces. TP-Spaces can be seen as a more generalized representation than all previous ones. C-Space, however, is the most generic and exact representation.

3 Trajectory Parameter Spaces (TP-Spaces)

In this section we first discuss the problem of measuring the distance to obstacles for a non-holonomic and any-shape robot, which is the basis for the definition of a TP-Space in section 3.2.

3.1 Distance-to-Obstacles

Calculating distance-to-obstacles (i.e. collision distances) is an essential step in any reactive navigation algorithm since it provides the robot with information for choosing the next movement. To the best of our knowledge, all previous works on reactive navigation make an implicit assumption that has never been questioned: distance-to-obstacles are computed by means of a single fixed path model: either circular or straight, depending on the robot having kinematic constraints or not. Distances are then taken along those 2D paths, though robot paths are actually defined as continuous sequences of locations and orientations, that is, as three-dimensional* curves in C-Space (see Fig. 6(a)). To be rigorous we propose to define distance-to-obstacles

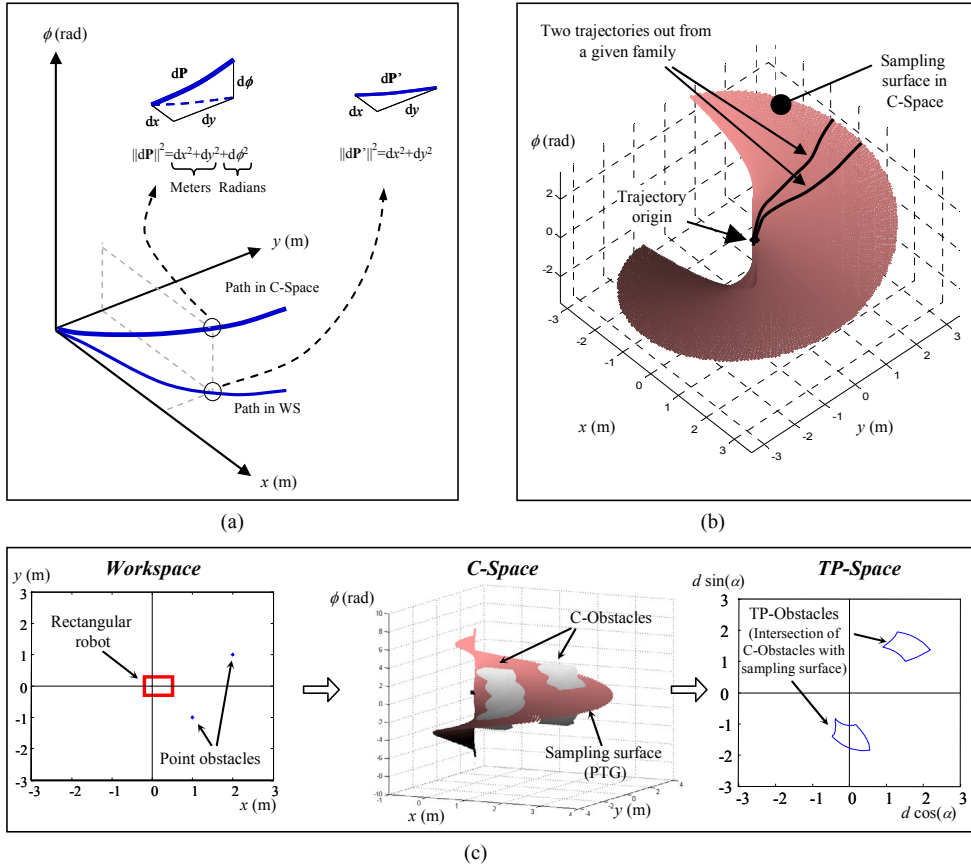


Fig. 6 Our vision of how to measure distance to obstacles. (a) Robot paths can be visualized as curves in C-Space. (b) A family of parameterized paths (PTG) generates a 3D surface that can be used to sample obstacles. (c) An example of this process to transform a set of sensed point obstacles into TP-Space, which can be seen as the intersection of the sampling surface with the C-Obstacles (see text for more details).

* We refer to C-Space as a 3D space due to the three dimensions of its topological structure $\mathbb{R}^2 \times S^1$, although this differs from the usual meaning of 3D Cartesian spaces with \mathbb{R}^3 structure.

directly in C-Space, through the mechanism that is described next.

If we visualize in C-Space all the paths from a given path model simultaneously we obtain a 3D surface, as the example in Fig. 6(b). We call these surfaces *sampling surfaces*, since distance-to-obstacle can be computed by measuring the distance from the origin (the current pose of the robot) to the intersection of those surfaces with C-Obstacles. Next we can “straighten out” the surface into a lower dimensionality space where the application of obstacle avoidance methods becomes practical. We will refer to the resulting 2D space as a TP-Space. Since we propose to use a diversity of path models simultaneously we will have different associated sampling surfaces in C-Space to compute distance-to-obstacles.

The above process is illustrated graphically with an example in Fig. 6(c). Notice that the measurement of distances in C-Space combines linear and angular values, as highlighted in Fig. 6(a). This problem can be worked out by defining alternative non-Euclidean metrics as we discuss later on.

3.2 Our definition of TP-Space

We define a TP-Space as any two-dimensional space where each point corresponds to a robot pose within a C-Space sampling surface. It is convenient to represent TP-Space points in polar coordinates: an angular component α and a distance d . The mapping between a TP-Space and a sampling surface is carried out by selecting an individual trajectory out from the family using the α coordinate, while d establishes the distance of the pose along that selected trajectory. In (Minguez and Montano 2006) such a distance is measured as the Euclidean distance disregarding the robot orientation. Alternatively, we propose to measure distances in TP-Space through a non-Euclidean metric, directly along C-Space sampling surfaces. This enables, for example, an unambiguous representation of pure rotation movements. Moreover, for convenience we normalize distances to the range $[0,1]$ for a sufficient large distance d_{MAX} , which settles the collision avoidance maximum foresee range. In other words, the region of interest in TP-Space is the circle of unit radius, that is, the range $A \times D \subset \mathbb{R}^2$, where $A = \{\alpha \mid \alpha \in]-\pi, \pi[\}$ and $D = \{d \mid d \in [0,1]\}$.

It must be kept in mind that the purpose of using TP-Space as a space representation is to convert the free-space detection problem for non-holonomic robots in C-Space into an easier

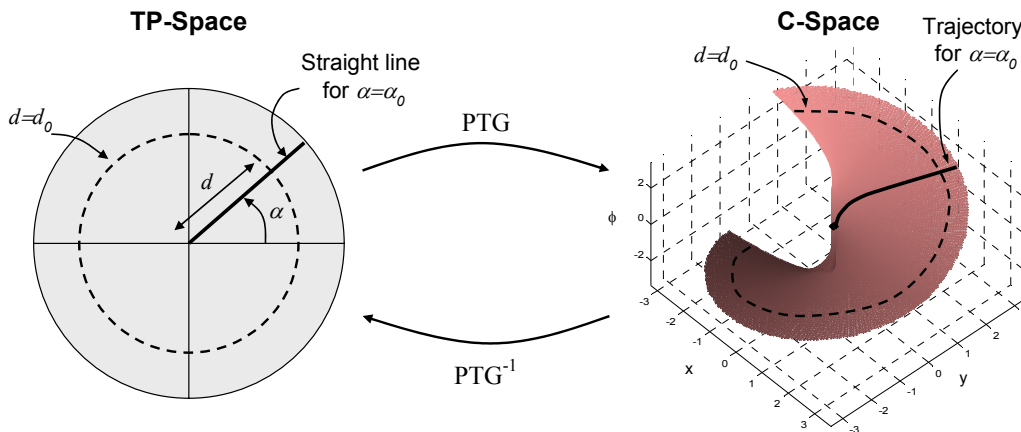


Fig. 7 A point given by its polar coordinates (α, d) in a TP-Space (left) represents a robot pose (x, y, ϕ) on a particular *sampling surface* in C-Space (right). The transformation is defined by a given PTG. The parameter α indexes a particular path within a family, while d indicates the distance from the origin.

collision avoidance problem in the transformed space. Note that the transformation is applied at each iteration of the navigation process, thus the robot is always at the origin while surrounding obstacles will be mapped into the unit circle.

4 Parameterized Trajectory Generators (PTGs)

4.1. Definition

A PTG is a mapping of TP-Space points (α, d) into 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. 7). Formally, a PTG is defined as:

$$\begin{aligned} \text{PTG: } A \times D \subset \mathbb{R}^2 &\rightarrow \mathbb{R}^2 \times S^1 \\ (\alpha, d) &\rightarrow \{(x, y), \phi\} \end{aligned} \quad (1)$$

where $A = \{\alpha \mid \alpha \in]-\pi, \pi[\}$ and $D = \{d \mid d \in [0, 1] \}$ define the domain of the PTG and S^1 represents the circular topology of the robot heading.

To map straight paths from TP-Space into paths feasible by the real robot in C-Space we propose to describe those paths in terms of *trajectories*: paths which also take time into account. Thus, linear and angular velocities are well defined at each point. Note that the time used in a PTG must not to be the robot actual time since it can be scaled for speed control purposes. Actually, time (t) is introduced in the PTG as a substitute of distances (d). 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 Euclidean metric for distances, since robot orientations are also involved. The change of variable therefore allows us to introduce a custom non-Euclidean metric in such a way that a one-to-one correspondence is established between distance and time. Before exposing the details of this custom metric, we need to introduce some definitions about trajectories.

Let $\mathbf{V}(\alpha, t) = [v(\alpha, t) \ \omega(\alpha, t)]^T$ be the velocity vector for a given trajectory, where its 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* in the context of a 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 computed by integration of the kinematic-constraint equation. For example,

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

stands for differential driven, tricycle, and car-like non-holonomic robots (Ramirez and Zegloul 2001). In general, this expression may not be integrable, thus an analytical expression for $\mathbf{P}(\alpha, t)$ will not be available for arbitrary design functions.

The problem of supplying a proper metric for C-Space can now be addressed. Mathematically, metrics are introduced through metric spaces, defined as tuples (E, ψ) where E is a *set* and ψ is a *distance* (function) between elements from the set. In our particular case, E is the C-Space and we define the distance as:

$$\begin{aligned} \psi: (\mathbb{R}^2 \times S^1) \times (\mathbb{R}^2 \times S^1) &\rightarrow \mathbb{R} \\ \psi(\mathbf{P}_1, \mathbf{P}_2) &= \left\| \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & r \end{bmatrix} \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \\ \phi_2 - \phi_1 \end{bmatrix} \right\| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (r \cdot (\phi_2 - \phi_1))^2} \\ \mathbf{P}_i &= [x_i \quad y_i \quad \phi_i]^T \in \mathbb{R}^2 \times S^1 \end{aligned} \quad (3)$$

where the change in the robot orientation is incorporated into the measurement by means of a given fixed distance r (e.g. the approximate radius of the robot). Notice that the requirements for any distance function hold for the above definition: (i) it gives non-negative values, (ii) a distance of zero is only obtained for $\mathbf{P}_1 = \mathbf{P}_2$, and (iii) it fulfills the triangle inequality. Our aim is to apply this metric to a given trajectory by measuring the distance from the origin to any given point along it. For this purpose it will be necessary to employ a distance function $m(\cdot)$ on infinitesimal pose increments:

$$m(d\mathbf{p}) \equiv \psi(\mathbf{p}, \mathbf{p} + d\mathbf{p}) = \left\| \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & r \end{bmatrix} \begin{bmatrix} dx \\ dy \\ d\phi \end{bmatrix} \right\| = \sqrt{dx^2 + dy^2 + (r \cdot d\phi)^2} \quad (4)$$

where $d\mathbf{p} = [dx \quad dy \quad d\phi]^T$. Since we are interested in normalized distances in the closed interval $[0, 1]$ for the TP-Space, we consider a normalization factor (d_{MAX}). If the normalized distance until time t is denoted as $\mu_\alpha(t)$ for a trajectory stated by α , and kinematics constraints are given by (2), then we obtain:

$$\begin{aligned} \mu_\alpha(t) &= \frac{1}{d_{MAX}} \int_0^t m\left(\frac{\partial \mathbf{P}(\alpha, \tau)}{\partial \tau}\right) d\tau = \frac{1}{d_{MAX}} \int_0^t \left\| \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & r \end{bmatrix} \begin{bmatrix} \cos \phi(\alpha, \tau) & 0 \\ \sin \phi(\alpha, \tau) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(\alpha, \tau) \\ \omega(\alpha, \tau) \end{bmatrix} \right\| d\tau \\ &= \frac{1}{d_{MAX}} \int_0^t \sqrt{v(\alpha, \tau)^2 + (r \cdot \omega(\alpha, \tau))^2} d\tau \end{aligned} \quad (5)$$

where the custom metric (3)-(4) has been used instead of the Euclidean norm. This function establishes a bijective mapping between time t and the normalized distance $\mu_\alpha(t)$, thus it solves the change of variable of PTGs. We finally arrive to a general expression for PTGs:

$$PTG(\alpha, d) = \mathbf{P}(\alpha, t'), \quad t' = \mu_\alpha^{-1}(d) \quad (6)$$

where $\mathbf{P}(\alpha, t)$ can be given from a closed-form expression or solved numerically, and the function $\mu_\alpha^{-1}(d)$ exists and is well-defined under the assumption that both design functions in $\mathbf{V}(\alpha, t)$ are not null simultaneously for any value of t .

We also define the inverse *PTG* function, which is involved in the construction of TP-Obstacles. It translates C-Space poses back into TP-Space, and is given by:

$$\begin{aligned} PTG^{-1}: Surf(PTG) \subset \mathbb{R}^2 \times S^1 &\rightarrow \mathbf{A} \times \mathbf{D} \\ ((x, y), \phi) &\rightarrow (\alpha, d) \end{aligned} \quad (7)$$

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

$$Surf(PTG) = \{\mathbf{P} \mid \mathbf{P} = PTG(\alpha, d)\} \quad \forall (\alpha, d) \in \mathbf{A} \times \mathbf{D} \quad (8)$$

About the units used for the distance parameter (d), it follows from our definition of TP-Space that the distance along straight paths in this space is directly stated by the d parameter, which in turn reflects the distance along a more complex trajectory in C-Space. Since our custom metric takes into account both the change in robot orientation and in the 2D location, strictly speaking we can not use a common distance unit (like meters) when referring to TP-Space distances, since rotations are also included in the measurement. We propose to use *pseudo-meters* (*psm*) as the

unit for distances in TP-Space, defined as the numerical value of (5) when the robot location is given in meters, and its orientation in radians. For referring to speeds in TP-Space we consequently use pseudo-meters per second (*psm/s*), the unit in which holonomic methods must give the desired robot speed in the virtual WS.

4.3 Design of PTGs

Not any arbitrary design function $V(\alpha, t)$ leads to a valid PTG, since it must be somehow guaranteed that collision avoidance can be performed in the resulting transformed space by the reactive method running on it. Those methods have been designed to work in the WS, but they will be applied to TP-Space (which can be seen as a virtual WS). Thus, several requirements are needed to assure some common assumptions about WS. Concretely, we define a *valid* PTG as that one fulfilling all the following requirements:

- It must generate *consistent reactive trajectories*. Any arbitrary path model is not applicable to reactive navigation because of the memoryless nature of the movement decision process (Blanco et al. 2005).
- It must be *WS-bijective*. No more than one trajectory can exist taking the robot from its current pose to any other WS location (x, y) , regardless of the orientation. Otherwise, the target position would be seen at two different directions (straight lines) in TP-Space – recall that a PTG maps straight lines of the TP-Space into trajectories of the C-Space.
- It must be *continuous*. Together with the last restriction, this condition assures that transformations do not modify the topology of the robot planar workspace.

Besides these restrictions, other considerations may be also taken into account when designing a PTG, such as the robot speed limit or any other kinematic constraints. For example, a car-like robot will impose a minimum turning radius, which becomes a maximum angular-to-linear velocity ratio.

While circular arcs (the classic path model) can be shown to fulfill the above requirements, a proof for the general case can not be provided due to the lack of a generic analytical solution for (2). In turn, we have designed *PTG templates* that guarantee all those conditions, that is, only valid PTGs are obtained from them. For our experiments we have designed five of these templates that cover a sufficiently wide diversity of trajectories, which are summarized in Fig. 8. The rest of this section provides a more detailed discussion of these templates.

A common feature to all of the following design schemes is that a variety of PTGs can be generated from each one by choosing different values of certain design parameters.

C PTG: *Circular trajectories*

This is the simplest path model, and the only one used in previous works on reactive navigation. Velocities remain constant with time along a given trajectory, as follows from the design equations in Fig. 8. In this case (2) is integrable and gives us the following closed form expression for trajectories:

$$\mathbf{P}(\alpha, t) = \begin{cases} \begin{bmatrix} \frac{Kv_0}{\omega_0 \tan \frac{\alpha}{2}} \sin\left(t\omega_0 \tan \frac{\alpha}{2}\right) \\ \frac{Kv_0}{\omega_0 \tan \frac{\alpha}{2}} \left(1 - \cos\left(t\omega_0 \tan \frac{\alpha}{2}\right)\right) \\ t\omega_0 \tan \frac{\alpha}{2} \end{bmatrix}, & \alpha \neq 0 \\ \begin{bmatrix} Kv_0 t \\ 0 \\ 0 \end{bmatrix}, & \alpha = 0 \end{cases}$$

where the parameter K is introduced for accounting with forward and backward trajectories (for values of 1 and -1 , respectively).

α -A PTG: *Trajectories with asymptotical heading*

These trajectories are generated by linear/angular velocities which are directly/inversely proportional to the difference between the robot heading and the parameter α . As a result, trajectories tend asymptotically to straight paths from the origin (see the corresponding diagram in Fig. 8). An example of a situation where a PTG from this template may be useful was already depicted in Fig. 2. For our robots moving in office-like scenarios, this is one of the most frequently selected PTG. Unfortunately, this PTG results in non-integrable trajectories and requires numerical solutions.

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

Trajectories for this template can be integrated to a closed form expression:

$$\mathbf{P}(\alpha, t) = [x(\alpha, t) \ y(\alpha, t) \ \phi(\alpha, t)]^T$$

$$x(\alpha, t) = \begin{cases} \frac{v_0 \left(-2\alpha\beta + 2\beta(\alpha - t\omega_0) \cos t\omega_0 + (4\pi^2 - \beta(\alpha^2 - 2t\omega_0\alpha + t^2\omega_0^2 - 2)) \sin t\omega_0 \right)}{4\pi^2 \omega_0}, & t \leq \left| \frac{\alpha}{\omega_0} \right| \\ x\left(\alpha, \left| \frac{\alpha}{\omega_0} \right|\right) + tv_0 \cos \alpha, & t > \left| \frac{\alpha}{\omega_0} \right| \end{cases}$$

$$y(\alpha, t) = \begin{cases} \frac{v_0 \left(4\pi^2 + \beta(2 - \alpha^2) + 2\beta(\alpha - t\omega_0) \sin t\omega_0 + (-4\pi^2 + \beta(\alpha^2 - 2t\omega_0\alpha + t^2\omega_0^2 - 2)) \cos t\omega_0 \right)}{\text{sign}(\alpha) 4\pi^2 \omega_0}, & t \leq \left| \frac{\alpha}{\omega_0} \right| \\ y\left(\alpha, \left| \frac{\alpha}{\omega_0} \right|\right) + tv_0 \sin \alpha, & t > \left| \frac{\alpha}{\omega_0} \right| \end{cases}$$

$$\phi(\alpha, t) = \begin{cases} t\omega_0, & t < \left| \frac{\alpha}{\omega_0} \right| \\ \alpha, & t \geq \left| \frac{\alpha}{\omega_0} \right| \end{cases}$$

where β is a design parameter in the range $[0,1]$ that slightly modifies the shape of trajectories. The main purpose of defining this template is to show that valid, consistent reactive trajectories exist with a closed form expression and which are not a composition of circular arcs and straight segments.

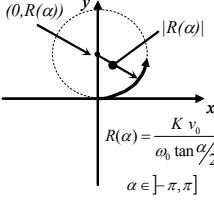
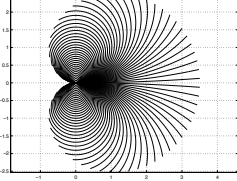
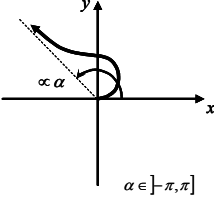
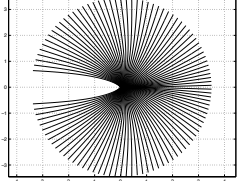
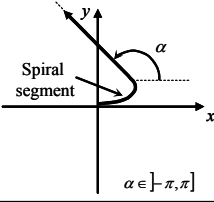
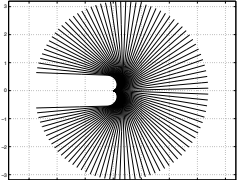
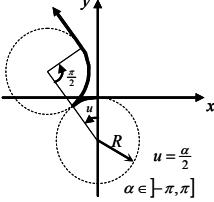
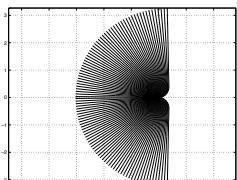
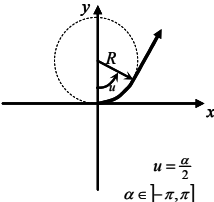
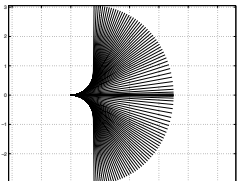
| PTG | Type of trajectory | PTG design equations | Design parameters | Example of generated paths |
|---------------------------|---|---|---------------------------|---|
| C |  | $\mathbf{V}(\alpha, t) = \begin{bmatrix} K v_0 \\ w_0 \tan \frac{\alpha}{2} \end{bmatrix}$ | v_0, w_0 $K = \pm 1$ |  |
|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

Fig. 8 Some examples of valid PTG design templates which have been evaluated on real robots. The table shows 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). Notice that the path model labelled as C is the only model considered by all previous works on reactive methods. 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.

C|C_{π/2}S and CS PTG: A set of optimal paths

These path models have been well studied in the field of motion planning and represent some of the optimal (shortest) path models for a car-like robot with a minimum turning radius R (Vendittelli et al. 1999). To the best of our knowledge, the present work is the first one to enable the incorporation of optimal solutions from the motion planning field into a reactive method. In this case both models lead to integrable expressions, although they are omitted for clarity since the results are straightforward concatenations of circular and straight segments. We must emphasize that using a reactive method does not assure the optimality of the robot path, thus the introduction of these PTGs implies that, only if the target is within the reactive foresee range and there are no obstacles along the optimal paths, the robot will actually follow a shortest path. Naturally, this statement cannot be extended to a general situation, although we believe it is still an important contribution in the scope of reactive navigation approaches.

4.4 Translating the Real Environment into TP-Space

Reactive navigation into TP-Space implies two transformations: (i) the construction of TP-Obstacles, and (ii) the translation of the target position (without orientation) into TP-Space. Although both transformations map 2D points from the environment into TP-Space, they are managed in quite different ways. The target location remains as a single point in TP-Space and it is computed at each iteration by the inverse PTG function in (7), ignoring the robot heading at the target location. Regarding the translation of obstacles, we will assume without loss of generality that only point obstacles exist. If any other kind of obstacles needs to be modelled (e.g. polygonal obstacles) they can be parameterized as a continuous sequence of point obstacles and TP-Obstacles built by composition. Moreover, in practice, most robotic sensors provide a point-like representation of the sensed obstacles. When point obstacles are moved into TP-Space each point becomes a region, named *TP-Obstacle*. 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 point \mathbf{o} as:

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

That is, TP-Obstacles are defined as the transformation of the intersection between C-Obstacles and the sampling surface of the PTG into TP-Space. Since a robot can collide with any obstacle from many different poses, *TP-Obstacles* are always two-dimensional regions in TP-Space. Holonomic obstacle avoidance methods will measure obstacle distances along straight paths in TP-Space, thus, in practice, only the closest obstacle must be kept for each direction of the transformed space. The whole process of transforming obstacles into TP-Space is illustrated in Fig. 9, where the last graph shows a polar plot of the closest TP-Obstacle at each direction as normalized distances. TP-Space charts in the rest of this paper will be shown using this polar representation.

The process of building TP-Obstacles implies computing only the part of C-Obstacles that contributes with useful information: the intersection with a given sampling surface. Although this is a computational expensive operation, an efficient method to obtain an approximate solution has been developed. It must be remarked that exact solutions may exist for *integrable* PTGs, but in this work we rely on an approximate method to enable the utilization of any PTG, regardless its integrability.

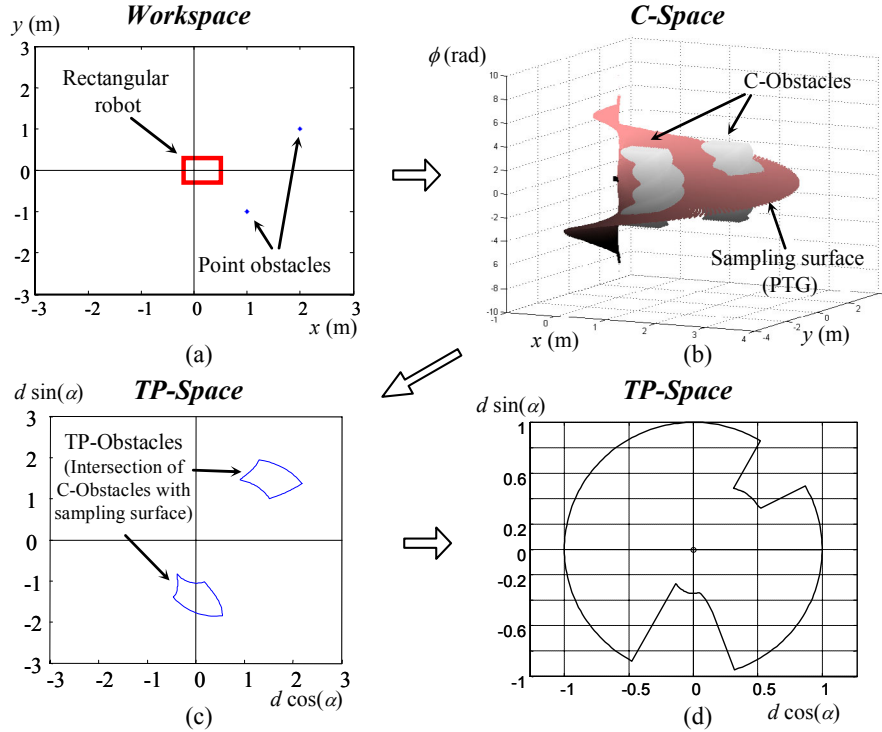


Fig. 9 The whole process from the sensed point-obstacles (a) to TP-Obstacles (c). The intersection of C-Obstacles with the sampling surface generated by a PTG (b) is projected into the TP-Space plane. We only keep the closest normalized obstacle-distance in TP-Space at each direction, as shown in (d), since this is the only information needed by obstacle avoidance methods. (For clarity, charts (a)-(c) that appeared in Fig. 6 are repeated here).

The method for computing the intersection of C-Obstacles with the PTG sampling surface is based on a lookup table with collision tests pre-calculated by simulation. A closely related idea was previously reported in (Schlegel 1998) for the case of circular paths and any-shape robots, although here the process is extended to an arbitrary path model given by a PTG: the physical space around the robot is arranged into a rectangular grid whose cells store their associated TP-Obstacle, built from the set of pairs (α, d) that lead to collision, as illustrated in Fig. 10. Therefore, the problem of translating a set of obstacles into TP-Space becomes that of adding the TP-Obstacles elements for the occupied cells. The grid must provide a sufficient resolution (e.g. 1 to 2cm) for the robot to detect narrow passages.

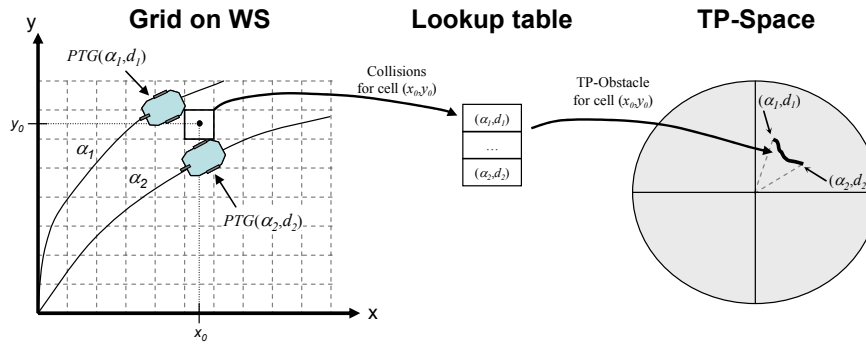


Fig. 10 The process for a fast construction of TP-Obstacles involves building a rectangular grid where each cell keeps the collision pairs (α, d) for its associated TP-Obstacle and a given PTG. Then moving real obstacles into TP-Space is accomplished by the addition of the occupied cells.

4.5 Transformation of Movement Commands Into Real World Actions

We consider that the motion generated by the holonomic collision avoidance method in the virtual WS (the TP-Space) is a pair stating the robot desired speed s (in psm/s units^{*}) and direction α_m (in the interval $]-\pi, \pi[$). This motion command is translated back into a real robot movement in two steps:

- We obtain a normalized velocity command as $\mathbf{V}_{norm}(\alpha_m) = \mathbf{V}(\alpha_m, 0) = [v(\alpha_m, 0) \ \omega(\alpha_m, 0)]^T$, through the evaluation of the PTG design functions at $\alpha = \alpha_m$. 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.
- The velocity command for the real robot \mathbf{V}_{rob} is computed by scaling \mathbf{V}_{norm} according to the holonomic velocity s in TP-Space (provided by the holonomic method). Mathematically:

$$\mathbf{V}_{rob}(\alpha_m, s) = \frac{s}{\max_{\alpha} \{m(\mathbf{V}(\alpha, 0))\}} \cdot \mathbf{V}_{norm}(\alpha_m) \quad (10)$$

where $m(\cdot)$ is the custom metric in (4), which in this case gives us:

$$m(\mathbf{V}(\alpha, 0)) = m\left(\frac{\partial \mathbf{P}(\alpha, \tau)}{\partial \tau}\right) = \left\| \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & r \end{bmatrix} \begin{bmatrix} \cos \phi(\alpha, 0) & 0 \\ \sin \phi(\alpha, 0) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(\alpha, 0) \\ \omega(\alpha, 0) \end{bmatrix} \right\| = \sqrt{v(\alpha, 0)^2 + (r \cdot \omega(\alpha, 0))^2} \quad (11)$$

5 The complete reactive navigation system

To exploit the advantages of navigation using TP-Spaces, we propose the navigation system sketched in Fig. 11, which is a detailed view of Fig. 3(b). The overall system comprises a control loop where sensor readings along with an estimation of the target relative location are supplied to the reactive navigation system. This system computes a velocity command that is sent back to the

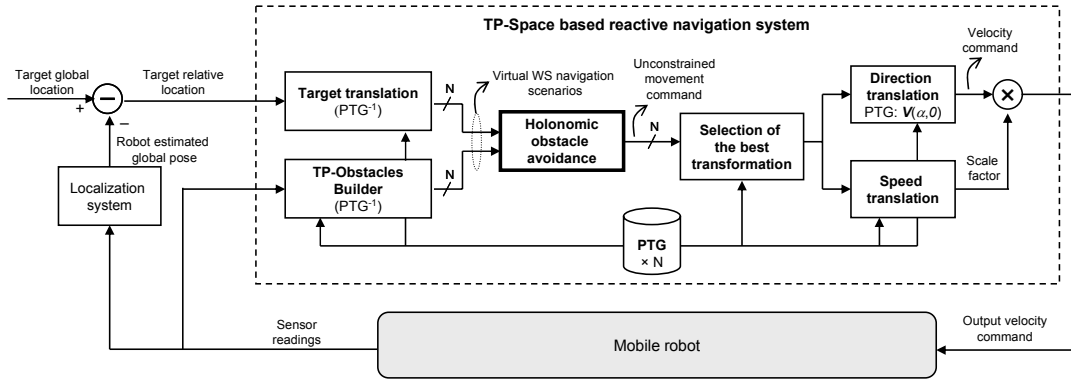


Fig. 11 A complete reactive navigation system based on TP-Space involves translating obstacles and the target location into the TP-Space, through a variety of PTGs simultaneously. Each one generates a virtual WS navigation scenario which is solved by a simple obstacle avoidance method. Next, the resulting movements are evaluated to find the most advantageous transformation, which is selected to generate the real robot velocity command using the PTG design equations.

^{*} As previously exposed, psm/s stands for pseudometers (psm) per second, where psm is the distance unit in TP-Space.

robot, closing the control loop. This process is repeated periodically (e.g. at 10Hz), resulting in a fast response to dynamic obstacles while steering the robot towards the target. We assume that the target location is given in some fixed coordinate system and that a localization system (not addressed here) is available to accurately estimate the position of the target relative to the robot.

Within the system, firstly the sensed obstacles and the target are translated into TP-Space, typically using several PTGs simultaneously and yielding a different virtual WS (in TP-Space) for each PTG. Next, any obstacle avoidance method suitable for a free-flying-point robot is applied to each virtual WS to obtain their respective movement commands. To select the most advantageous one we evaluate each of the potential movements by weighting the following factors* normalized within the range [0,1]:

f_1 : Collision-free distance for the selected movement direction (in TP-Space).

f_2 : Relative direction to target in TP-Space. Let α_m be the desired movement direction, and α_t the direction towards the target, in the transformed space. Then:

$$f_2 = e^{-\left(\frac{\alpha_m - \alpha_t}{\frac{2}{3}\pi}\right)^2} \quad (12)$$

f_3 : Robot heading towards target in C-Space. This term prioritizes the movements that make the robot to face the target, in the real world. If θ is the heading of target at the end of robot direction movement, then this factor can be evaluated as:

$$f_3 = e^{-\left(\frac{\theta}{\pi/2}\right)^2} \quad (13)$$

f_4 : Euclidean distance to the target (in WS). This factor gives more relevance to movements that take the robot closer to the target:

$$f_4 = \frac{d_{\max} - \min\{d_t, d_{\max}\}}{d_{\max}} \quad (14)$$

where d_t is the minimum distance from robot to target in the selected trajectory and d_{\max} is the normalization factor specified in the PTG design.

f_5 : Hysteresis. This factor contributes to stabilize the behavior of the system, avoiding frequent oscillations between different possible transformations that perform similarly through short periods of time:

$$f_5 = \begin{cases} 1, & \text{if this PTG was selected in the last iteration.} \\ 0, & \text{otherwise.} \end{cases} \quad (15)$$

Logically, the global behavior of the system depends on the values of these factors, though empirically we have determined that a rough tuning suffices to achieve a good performance in most situations and on different robots. In our implementation we have applied the weights {0.4, 0.15, 0.15, 0.2, 0.1} for each factor, respectively. Intuitively, these distribution of the weights gives a maximum relevance to the clearance (term f_1), and in second place to the closeness of the trajectory toward the target (term f_4). The transformation with the highest score is chosen at each iteration, and next its corresponding movement is converted back into a velocity command for

* Note that similar heuristic factors have been proposed in most V-Space approaches, e.g. Arras et al. 2002 and Simmons 1996.

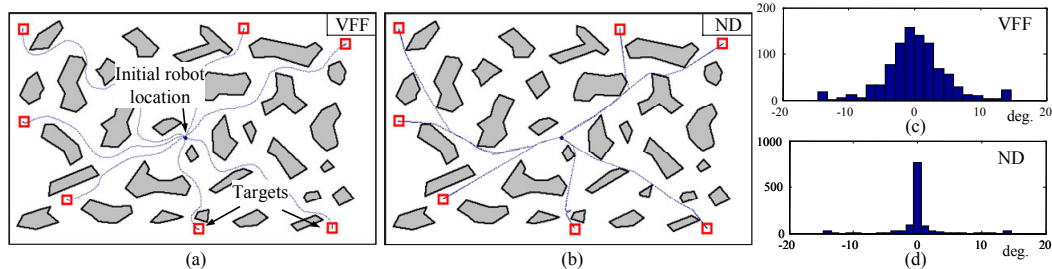


Fig. 12 A comparison between two implemented holonomic obstacle avoidance algorithms in a synthetic environment: (a) and (b) show navigations from a central point towards seven different target locations using the Virtual Force Field (VFF) and the Nearness diagram (ND) methods, respectively. The later generates straighter paths for the virtual holonomic robot, which makes the real robot less shaky in cluttered environments. This is confirmed by the histograms of the changes in the steering between iterations, shown in (c) and (d) for both methods.

the real robot as discussed in a previous section: first, a normalized command is directly extracted from the PTG design equations, and then it is scaled to account for the speed given by the obstacle avoidance method.

We have experimented with two holonomic obstacle avoidance methods in TP-Space: a Virtual Force Field (VFF) method and a custom implementation of the Nearness Diagram approach (Minguez and Montano 2004). To compare the performance of these methods on a kinematically-free robot, we have carried out some simulated experiments whose results are shown in Fig. 12. A major distinctive feature of ND is that it results in mostly straight paths, which is reflected in few changes in the real robot direction (see the histograms of Fig. 12(c)-(d)). Nevertheless, VFF is simpler to implement than ND, and may perform appropriately in uncluttered environments.

The major concern in getting a real-time functional implementation of this navigation system is to account for a fast TP-Obstacles building process, which has been achieved through the efficient procedure based on precomputed lookup-tables that was previously discussed. One lookup-table is stored for each PTG for a fast consultation during navigation. Although building these tables takes a long time (typically 15-35 seconds for our configuration), they must be updated only if the shape of the robot changes; on the other hand, they allow TP-Obstacles to be built in such a short time as 0.2ms per PTG. This short transformation time and the quick response of the holonomic methods keep the whole system fast enough to perform in real-time within dynamic scenarios.

6 Results and Discussion

TP-Space based navigation has been intensively tested for more than a year in office-like scenarios with our robots *SENA*, a robotic wheelchair (Fernández-Madriral et al. 2004, Gonzalez et al. 2006), and *Sancho*, a service robot built upon a Pioneer 3-DX mobile base, both driven differentially and equipped with laser range finders for obstacle detection. Next we discuss three different experiments, each one aimed to illustrate a differentiated feature of the proposed navigation system. We encourage the reader to also consider watching the online videos corresponding to these experiments in <http://www.isa.uma.es/C3/C1/Navigation/>.

6.1 First experiment: Simulated robot

This experiment demonstrates the feasibility of our approach for dealing with a car-like robot in simulated cluttered scenarios. It must be remarked that any reactive method that ignores the robot

kinematics will fail if applied to a kinematically-constrained robot. The car-like robot is commanded to navigate towards the target point shown in Fig. 13(a) making use of two PTGs: $C|C_{\pi/2}S$ and CS, both being able to generate (possibly) optimal paths for this kind of robot (Vendittelli et al. 1999). The selection of the active PTG at each instant of time is graphically represented in Fig. 13(b), where it can be seen how the $C|C_{\pi/2}S$ model is solely used when maneuvering to get the robot out of the starting location. Notice how our approach allows such a *maneuvering* in a *memoryless* system. Nevertheless, it must be stressed that any reactive system, including the one described in this work, has limited foreseeing capabilities and may not be able to find a valid path in all the situations.

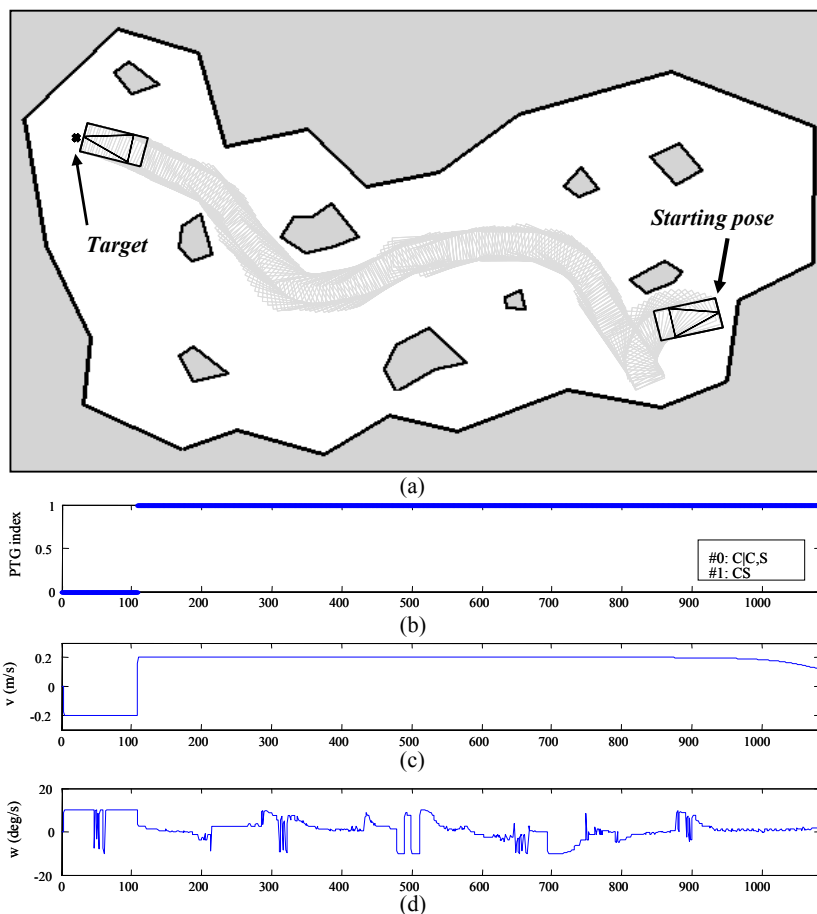


Fig. 13 A simulated experiment to demonstrate the manoeuvring capabilities of a car-like robot with our reactive navigation system. (a) The resulting trajectory, (b) the active PTG at each instant of time, and (c)-(d) the linear and angular velocities of the robot.

6.2 Second experiment: Robotic Wheelchair

In this experiment with the robotic wheelchair *SENA* (González et al. 2006) we illustrate the feasibility of controlling a kinematically constrained real mobile platform with a strong rectangular-like shape using a simple obstacle avoidance method (ND in this case). Ignoring the shape of this robot would most probably lead to collision. The resulting trajectory for this experiment is shown in Fig. 14(d), and some snapshots during the navigation can be seen in Fig. 14(e)-(h). Here we have employed four simultaneous PTGs: a forward C PTG, a backward C PTG, and two different instances of the α -A PTG, whose selection over time is plotted in Fig. 14(a). In this experiment the backward movement (PTG with index 1) is never selected.

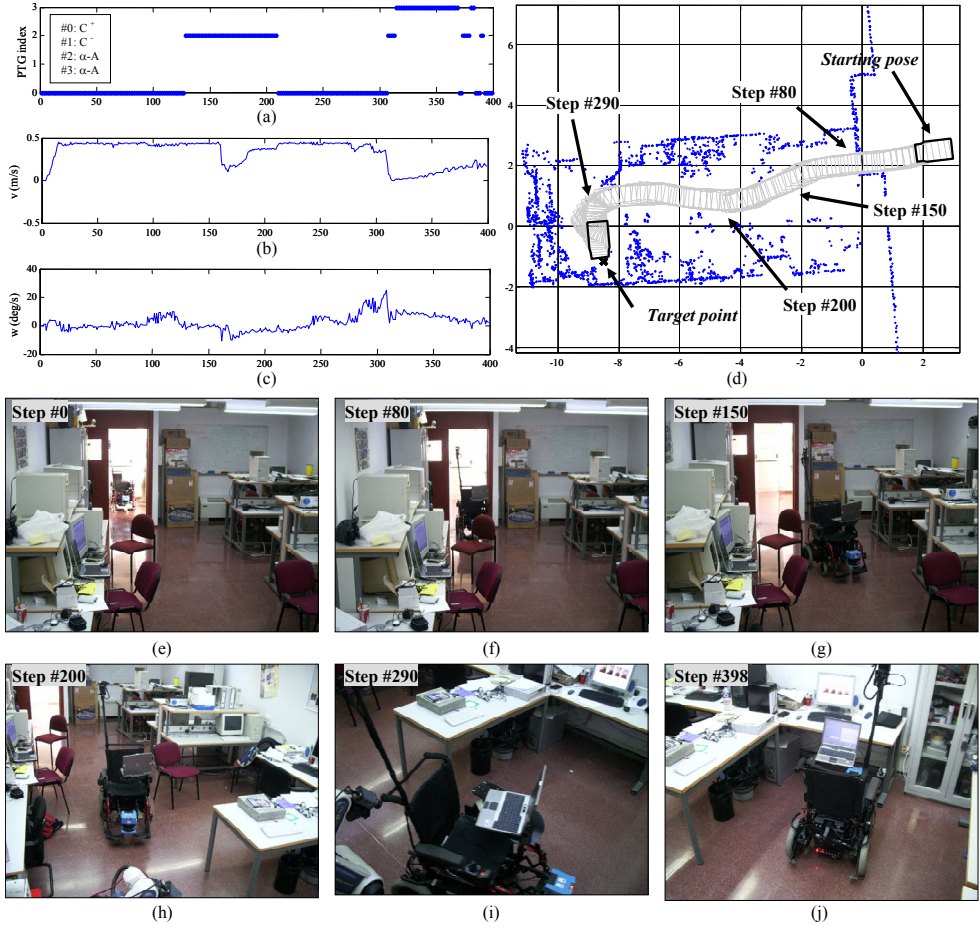


Fig. 14 In this experiment, SENA (a robotic wheelchair) performs a reactive navigation using four PTGs, whose selection over time is shown in (a). The linear and angular velocities along the navigation are plotted in (b)-(c), respectively, and the resulting trajectory is depicted in (d). Snapshots (e)-(j) show some instants along the course of the navigation.

Regarding the time requirements of the system for this configuration of PTGs, it takes a mean of 1.22ms (on a 1.8GHz Pentium-M) to execute one complete step of the navigation system, including the mapping of all the obstacles into TP-Space, ND-based collision avoidance, the selection of the best movement, and its transformation into a command for the real robot. Provided that the execution frequency is 10Hz, our method occupies a mere 1.22% of the CPU time.

6.3 Third experiment: Comparison to the “arc approach”

It is worth to illustrate a situation where our approach could be compared to previous reactive techniques in order to reveal its advantages. The following experiment consists of our mobile robot *Sancho*, equipped with a front and a rear laser scanner, being commanded a reactive navigation from a cluttered room towards a corridor outside. The navigation is repeated twice: the first time the robot uses a set of five PTGs (of the types C, α -A, and CS), while the second time it uses just forward and backward circular arcs (type C PTG). The overall trajectories of the robot for each case are shown in the two graphs at the top of Fig. 15. In this cluttered environment the obstacle avoidance method (the ND algorithm) has problems finding a good movement by using just circular arcs. One can clearly observe a much more poor performance of

the navigation in this case, in contrast to the case of the five PTGs, as also revealed by the overall path length and time taken by each navigation, plotted in Fig. 15. It is remarkable that by using more path models (five PTGs) the robot accomplishes the navigation in 88 seconds less than using just circular arcs, also saving 11.4m from the overall path length. One of the reasons of these differences is that, for the case of the circular arcs, the robot encountered significant problems at some specific places along its trajectory, such as turning a pronounced corner of about 90° or passing the doorway by the end of the navigation. For comparison purposes, both events are marked (with their respective time steps 241 and 591) in Fig. 15 for the path described using five PTGs. To illustrate how the additional PTGs have improved the navigation of the robot at those specific parts of the navigation, in that figure we show the sensed obstacles and their translations into TP-Space for the PTGs selected at those time steps. At time step 241, a PTG of the type CS (refer to section 4.3) is selected since it includes a movement which fits perfectly to the maneuver the robot needs to bypass the obstacle. The narrow “gap” in the TP-Obstacles which appears in this case represents the small range of CS trajectories (α parameter values) that makes the robot to pass by the corner without colliding. Similarly, the graphs for time step 591 show how a PTG of the type α -A reveals collision-free maneuvers for passing through the narrow doorway. We must emphasize that these PTGs are selected at these specific instants of time because the circular arcs are not able to find better movements using the criterion of the obstacle avoidance algorithm.

To sum up, this experiment has described an example of how the use of other path models apart from arcs introduces a significant improvement into the overall performance of the navigation. Despite the fact that circular arcs may be the optimal choice for the most part of a typical navigation, the existence of other choices at some specific moments allows the robot to find maneuvers that would be otherwise much more difficult (or even impossible) to find by means of arcs, a fact that ultimately may determine the performance and reliability of the whole reactive navigator.

6.4 Fourth experiment: Dynamic environments

Finally, we present an experiment where SENA, the robotic wheelchair, navigates for almost two hours in a non-prepared, crowded scenario. We have chosen for this experiment the entrance of our building at the University of Málaga (Spain). The robot is commanded to navigate repeatedly between two fixed target locations, labeled as ‘A’ and ‘B’ in Fig. 16(a)-(b), and in its way it must avoid some static obstacles (columns, furniture, etc.) and moving people. During the experiment the robot avoids collisions with several people walking in the opposite direction than the robot, as well as some students who intentionally try to block its way. In all the cases, the reactive

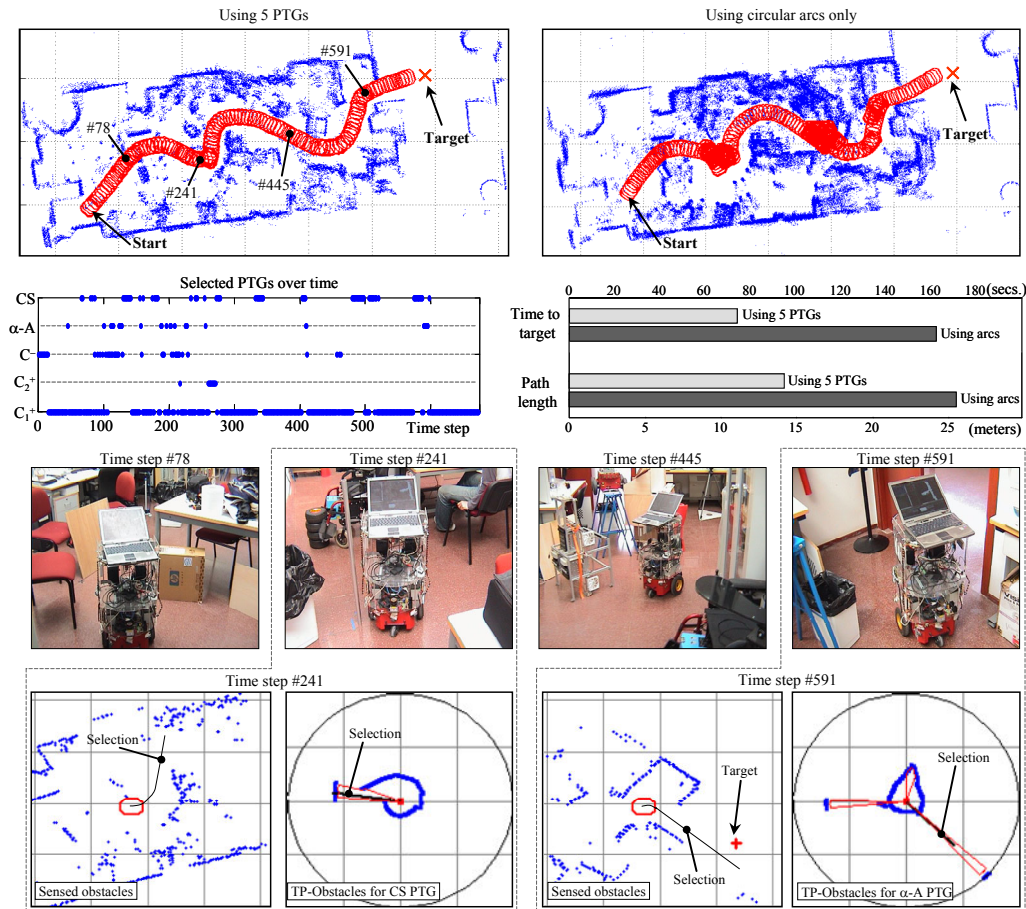


Fig. 15. A comparison of performance between the same obstacle avoidance method using five PTGs and only circular (forward and backward) arcs. It is shown how the use of several PTGs improves both the time consumed by the navigation and the overall path length. Some snapshots along the navigation show the cluttered scenario and the sensed obstacles at some specific instants of time, both in the robot workspace and in the TP-Space using the PTG selected at each time. See the text for further discussion.

navigator (using five PTGs in this case) successfully avoids all the collisions and gets out of the blocking situations.

To perform this experiment in such a dynamic scenario, we have required robust localization within the environment, which has been carried out through standard particle filter techniques (Thrun et al. 2001) for occupancy grid maps (see Fig. 16(a)). For robustness both in localization and in detecting all the surrounding obstacles, we instrumented the robotic wheelchair with three laser range scanners at different heights and positions: one at the front, a rear scanner, and another one on the top of the robot, as highlighted in Fig. 16(e).

6.5 Conclusions

We have presented a new approach for the reactive navigation of a kinematically-constrained and any-shape mobile robot, on the basis of a clear and useful separation of the problems of robot shape and kinematic restrictions, and collision avoidance. For that, we have developed a generalized kinematics abstraction mechanism which allows us using a variety of path models (PTGs) to obtain a better sampling of the whole C-Space from which more and better collision-free paths towards the target location can be found. We have also shown how the generalization of path models enables the introduction of optimal path models into our reactive navigation system for the first time in a non-planned approach to robot navigation. The implementation of

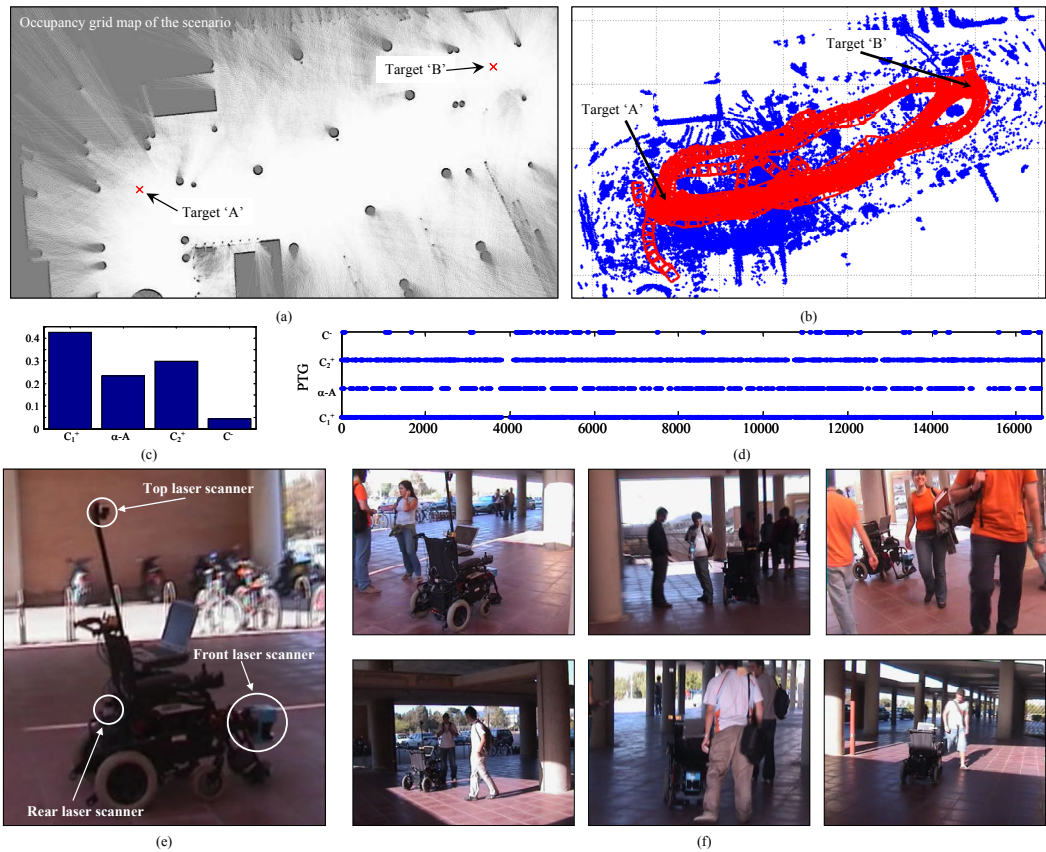


Fig. 16. In this experiment SENA navigated for two hours in a non-prepared scenario amidst dozens of students without colliding. (a) An occupancy grid map of the scenario. (b) Overlap of laser scans during the mission, which consists of going to a pair of targets sequentially. (c) Ratio of selection for each PTG, and (d) their selection over time. (e) Positions of the three laser scanners on the robot. (f) Some snapshots of the experiment.

the navigation system has demonstrated to be effective and very efficient for robots in cluttered, dynamics scenarios, which has been verified along more than two years of extensive tests. Future research lines include the introduction of robot dynamics into PTG and the integration of this reactive system with global planning.

Acknowledgements

This work has been supported by the Spanish Government under contract CICYT-DPI2005-01391. The authors want to thank J. Minguez for fruitful discussion during the preparation of this work.

References

- Arkin R.C. 1998. Behaviour-Based Robotics. MIT Press.
- Arras K.O., Persson J., Tomatis N., Siegwart R., 2002. Real-Time Obstacle Avoidance For polygonal Robots With a Reduced Dynamic Window. In IEEE Int. Conf. on Robotics and Automation, vol. 3, pp. 3050–3055.
- Balch T., Arkin R., 1993. Avoiding the past: a simple but effective strategy for reactive navigation. In IEEE Int. Conf. on Robotics and Automation, vol.1, pp. 678–685.
- Blanco J.L., Gonzalez J., Fernández-Madrigal J.A., 2005. The TP-Space: Foundations and applications. Technical Report, System Engineering and Automation Dept., University of Malaga.
- Borenstein J., Koren Y. 1989. Real-time Obstacle Avoidance for Fast Mobile Robots. In IEEE Transactions on Systems, Man, and Cybernetics, vol. 19, no. 5, pp. 1179–1187.
- Borenstein J., Koren Y. 1991. The Vector Field Histogram – Fast Obstacle Avoidance for Mobile Robots. In IEEE Transactions on Robotics and Automation, vol. 7, no. 3, pp. 278–288.
- Feiten W., Bauer R., Lawitzky G. 1994. Robust Obstacle Avoidance in Unknown and Cramped Environments. In IEEE Int. Conf. on Robotics and Automation, vol. 3, pp. 2412–2417.
- Fernández-Madrigal J.A., Galindo C., González J. 2004. Assistive Navigation of a Robotic Wheelchair using a Multihierarchical Model of the Environment. In Integrated Computer-Aided Engineering, vol. 11, no. 4, pp. 309–322.
- Fiorini P. and Shiller Z. 1998. Motion Planning in Dynamic Environments using Velocity Obstacles. In The International Journal of Robotics Research, vol. 17, no. 7, pp. 760–772.
- Fox D., Burgard W., Thrun S. 1997. The dynamic window approach to collision avoidance. In IEEE Robotics and Automation Magazine, vol.4, pp. 23–33.
- González J., Galindo C., Blanco J.L., Muñoz A.J., Arevalo V. and Fernández-Madrigal J.A. 2006. The Robotic Wheelchair SENA Project. In DAAAM International Scientific Book 2006, ch. 20, ISSN 1726-9687.
- Haddad H., Khatib M., Lacroix S., Chatila R. 1998. Reactive navigation in outdoor environments using potential fields. In IEEE Int. Conference on Robotics and Automation, vol.2, pp.1232–1237.

- Khatib, M., Jaouni H., Chatila R. and Laumond J.P. 1997. Dynamic Path Modification for Car-Like Nonholonomic Mobile Robots. In IEEE Int. Conf. on Robotics and Automation (ICRA), vol.4, pp. 2920–2925.
- Lamiroux F., Bonnafous D., Lefebvre O. 2004. Reactive Path Deformation for Nonholonomic Mobile Robots. In IEEE Transactions On Robotics, vol. 20, no. 6, pp. 967–977.
- Latombe C. 1991. Robot Motion Planning. Kluwer Academic Publishers.
- Laumond J.P., Souères P. 1993. Metric induced by the shortest paths for a car-like mobile robot. In IEEE Int. Conf. on Int. Robots and Systems, vol. 2, pp. 1299–1304.
- Lozano-Pérez, T. 1987. A Simple Motion-Planning Algorithm for General Robot Manipulators. In IEEE Journal of Robotics and Automation, vol. 3, no. 3, pp. 224–238.
- Minguez J., Montano L. 2004. Nearness Diagram (ND) Navigation: Collision Avoidance in Troublesome Scenarios. In IEEE Transactions on Robotics and Automation, vol.20, no.1, pp.45–59.
- Minguez J., Montano L. 2006. Abstracting vehicle Shape and Kinematics constraints from Obstacle Avoidance Methods. In *Autonomous Robots*, vol. 20, no. 1, pp. 43–59.
- Minguez J., Montano L. and Santos-Victor J. 2002. The Ego-Kinematic Space (EKS): Shape and Kinematics of the Vehicle. In IEEE International Conference on Robotics and Automation (ICRA), vol. 3, pp. 3074–3080.
- Murphy R.R., 2000. *Introduction to AI Robotic*. The MIT Press.
- Quinlan S. and Khatib O. 1993. Elastic bands: connecting path planning and control. In IEEE International Conference on Robotics and Automation (ICRA), vol. 2, pp. 802–807.
- Ramirez G., Zegloul S. 2001. Collision-free path planning for non-holonomic mobile robots using a new obstacle representation in the velocity space. In *Robotica*, vol. 19, pp. 543–555.
- Reeds J.A. and Schepp R.A. 1990. Optimal paths for a cat that goes both forward and backward. *Pacific Journal of Mathematics*, vol. 145, no. 2, pp. 367–393.
- Schlegel C. 1998. Fast Local Obstacle Avoidance under Kinematics and Dynamic Constraints for a Mobile Robot. In IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, vol.1, pp.594–599.
- Simmons R. 1996. The Curvature-Velocity Method for Local Obstacle Avoidance. In IEEE International Conference on Robotics and Automation (ICRA), vol. 4, pp. 3375–3382.
- Souères P., Laumond J.P. 1996. Shortest Paths Synthesis for a Car-like Robot. In IEEE Trans. on Autom. Control, vol. 41, no. 5, pp. 672–688.
- Thrun S., Fox D., Burgard W., Dellaert F., 2001. Robust Monte Carlo localization for mobile robots. In *Artificial Intelligence*, vol. 128, no. 1–2, pp. 99–141.
- Vendittelli M., Laumond J.P. and Nissoux C., 1999. Obstacle Distance for Car-Like Robots. In IEEE Transactions on Robotics and Automation, vol.15, no.4, pp. 678–691.
- Pal P.K., Kar A., 1995. Mobile Robot Navigation Using a Neural Net. In IEEE Int. Conf. on Robotics and Automation, vol.2, pp.1503–1508.
- Xu H., Yang S.X. 2002. Real-time collision-free motion planning of non-holonomic robots using a neural dynamics based approach. In IEEE International Conference on Robotics and Automation (ICRA), vol.3, pp. 3087–3092.
- Zhang L., Kim Y.J., Varadhan G., Manocha D. 2006. Fast C-Obstacle Query Computation for Motion Planning. In IEEE Int. Conference on Robotics and Automation (ICRA), pp. 3035–3040.