Proceedings of the 2002 IEEE
International Conference on Robotics & Automation
Washington, DC • May 2002

Reactive Navigation for Non-holonomic Robots using the Ego-Kinematic Space

Javier Mínguez[1], Luis Montano[1], José Santos-Victor[2]

[1]Dept. of Computer Science and Systems Engineering, University of Zaragoza (Spain). (jminguez,montano)@posta.unizar.es

[2]Instituto de Sistemas e Robótica, Instituto Superior Técnico (Portugal). jasv@isr.ist.utl.pt

Abstract

We address the problem of applying reactive navigation methods to non-holonomic robots. Rather than embedding the motion constraints when designing a navigation method, we propose to introduce the robot's kinematic constraints directly in the spatial representation. In this space - the Ego-Kinematic Space *- the robot moves as a "free-flying object". Hence, standard reactive navigation methods applied to this space will automatically take into account the robot's kinematic constraints, without additional modifications. This methodology can be used with a large class of constrained mobile platforms (e.g. differential-driven robots, car-like robots, tri-cycle robots). We show experiments involving non-holonomic robots with two reactive navigation methods whose original formulation does not take the robot kinematic constraints into account (the Nearness Diagram Navigation and a Potential Field method).*

1 Introduction

In this paper we address the problem of reactive navigation for non-holonomic robots. Even though the majority of robots exhibit kinematic constraints, most reactive navigation methods do not take these constraints into account and treat the robot as a "free-flying object". Example of these methods include, the Potential Field Methods [1], the Vector Field Histogram [2], the Elastic Band [7], the Elastic Strips [11], and the Nearness Diagram Navigation [12]. With these methods, the robot can only approximately execute the motion generated by the reactive schemes.

In other reactive navigation methods, the robot motion constraints are used directly in the process of determining the robot motion. However, with this approach each reactive navigation method must be designed from scratch to incorporate the constraints of a given robot. This process does not give general solutions to take into account the kinematic constraints, and it only provides particular solutions for each method. Examples of such methods include the Dynamic Window Approach [6], the Curvature Velocity method [5], the Vector Field Histogram+ [2], the Steering Angle Approach [4], and the non-holonomic Elastic Band [8].

Instead, we propose to use the kinematic constraints to construct a novel spatial representation, the *Ego-Kinematic Space*, where the robot moves as a "free-flying object". Hence, standard reactive methods can be applied in this space, implicitly taking into account the kinematic constraints. This methodology provides a general solution to apply (off-the-shelf) reactive navigation methods to non-holonomic robots.

We show that the *Ego-Kinematic Transformation* can be used to transform either the robot Workspace or (a simplification of) the Configuration space, as both can be represented with the same structure ($\mathbb{R}^2$) in the context of reactive navigation. Thus, the *Ego-Kinematic Space* can be used with most reactive navigation methods.

As an example, we use this methodology to generalize two reactive schemes that do not take into account the kinematic constraints (the Nearness Diagram Navigation [12], and a Potential Field method [1]), to work with non-holonomic robots.

The paper is organized in five sections: In Section 2 we introduce the kinematics of the case study robots; we discuss the role of reactive navigation methods and some properties of the Workspace and Configuration space. The *Ego-Kinematic Space* is described in detail in Section 3. In Section 4 we present experiments with two reactive navigation methods, and in Section 5 we draw our conclusions.

0-7803-7272-7/02/$17.00 © 2002 IEEE

2 Background & Preliminaries

In this section we introduce and review some concepts that will be used in the remaining part of the paper. We begin by presenting the robot's kinematics that will be covered in our analysis. Second, we discuss the technique used to generate the collision-free motion (the reactive navigation methods), and the spaces where these methods usually apply. Finally, the robots' kinematic model is used to derive some properties of the spaces where the reactive methods apply.

2.1 Robot's Kinematics

We consider robots moving on a flat surface with the classical hypothesis "rolling without slipping". The Workspace and Configuration space are given by $\mathcal{W} = \mathbb{R}^2$ and $\mathcal{C} = \mathbb{R}^2 \times S^1$, respectively. The robot configuration is represented by its location and the orientation $\mathbf{q} = (x, y, \theta)$. We shall focus our attention on robots whose motion is constrained by:

$$-\dot{x}sin\theta + \dot{y}cos\theta = 0 \qquad (1)$$

This is a non-holonomic equality constraint, which has the effect of reducing by one the dimension of the space of possible differential motions at any given configuration [9]. Hence, a motion command of such mobile robot can be fully described by two motion parameters only.

We now briefly describe the kinematic model of the *two-driving wheels* and the *car-like* mobile robots. These robots are representative examples of mobile platforms (see Fig. 1) that verify the constraint of Equation (1). We refer to [10] for a deeper description of both models.

The kinematic model of both, the two-driving wheels and car-like robots, can be expressed by the following equation by applying a change of variable, as detailed in [10]:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} cos\theta \\ sin\theta \\ 0 \end{pmatrix} v + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} w \qquad (2)$$

where v and w denote the linear and angular velocities.

The main difference between these two platforms is that the car-like robot has a mechanical constraint, which imposes a maximum curvature γ_{max} (or minimum turning radius $r_{min} = 1/\gamma_{max}$) of the path executed by the robot. There are other robots (e.g. tri-cycle robots) whose kinematic models can also be expressed by Equation (2) and verify Equation (1).

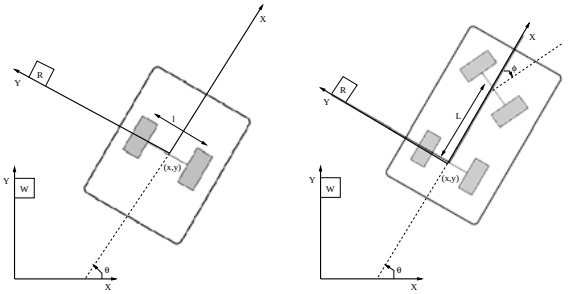

Figure 1: a)Two-driving wheels robot. b)Car-like robot.

2.2 Reactive Navigation Methods

To address the goals of this paper, we turn to a discussion of the role of reactive navigation methods in generating collision-free motion.

Reactive navigation methods attempt to calculate a motion command that generates a collision-free motion for the next sampling period T, while simultaneously driving the robot towards the goal. The result of applying iteratively a reactive navigation method is a sequence of motion commands that move the robot from the initial location towards the final location, while avoiding collisions.

In general, reactive navigation methods can be applied either to the *Workspace* (e.g. the Vector Field Histogram [2], the Dynamic Window Approach [6], the Steering Angle Field Approach [4], the Curvature Velocity Method [5], the Elastic Strips [11]) or to the *Configuration space* [9] (e.g. the Potential field methods [1], the Vector Field Histogram+ [3], the Elastic Band [7]).

2.3 Properties of the Workspace and Configuration space

The approach described in this paper consists of introducing a spatial transformation - prior to the application of a reactive navigation method - such that the robot kinematic constraints are directly represented. This transformation must be applied to the robot Workspace or Configuration space, depending on the class of navigation method used.

Based on the kinematic models of the robots under study, we will show that, within the context of reactive navigation, both the Workspace and a subset of the Configuration space - *the Reachable Set of a single Motion Command* - are fully represented by $\mathbb{R}^2$. Moreover, as far as the goal is to avoid obstacles, we

show that the obstacle information in both spaces is also completely represented by $\mathbb{R}^2$. Hence, a transformation defined over $\mathbb{R}^2$ has full generality.

As the Workspace is readily defined as $\mathbb{R}^2$, we will focus on the Configuration space. The Configuration space, $\mathcal{C} = \mathbb{R}^2 \times S^1$, describes the robot position and orientation. In the context of reactive navigation we only need to consider those paths in the Configuration space that can be obtained as the consequence of the execution of one single motion command. The set of all the configurations reachable by such paths define a subset of the Configuration space - the *Reachable Set of a single Motion Command* - that we will denote by $\mathcal{R}^{1mc}$.

Naturally, the structure of $\mathcal{R}^{1mc}$ will depend on the kinematic model of a specific mobile robot, subject to its own motion constraints. As in [6] we assume that, for the robots discussed in Section 2.1, the trajectories obtained by the execution of a single motion command can be approximated by arcs of circle and line segments. When the robot is on a circular path, its orientation at each point is constrained by:

$$tan\theta - \frac{2xy}{y^2 - x^2} = 0 \tag{3}$$

which is expressed in the robot's frame of reference.

This equation is a holonomic equality constraint, which has the effect of reducing the dimension of the Configuration space by one [9]. Therefore, the Reachable Set of a single motion command $\mathcal{R}^{1mc}$ (i.e. the set of all the configurations that can be reached by a straight segment or an arc of a circle) is a function of only two parameters, and can be represented by $\mathbb{R}^2$. A configuration in $\mathcal{R}^{1mc}$ is expressed by $\mathbf{q^{1mc}} = (x, y)$.

A point worth mentioning here is the fact that both the robot Workspace, $\mathcal{W}$, and the Reachable Set of a single Motion Command, $\mathcal{R}^{1mc}$, can be described in $\mathbb{R}^2$.

For reactive navigation we still need to map the obstacles, that are usually defined in the Workspace $\mathcal{W}$, into the Reachable Set of a single Motion Command $\mathcal{R}^{1mc}$.

We assume that obstacles are represented in $\mathcal{W}$ by a set of points $(x, y) \in \mathbb{R}^2$. Each obstacle point is mapped to a region in $\mathcal{R}^{1mc}$, named *C-obstacle* [9]. Details of this simple algorithm are beyond of the scope of this paper, but let us note in passing that this region is calculated by considering the paths induced to reach a given point, as well as the shape of the robot. The union of the C-obstacles is called the C-obstacle region, $\mathcal{R}^{1mc}_{obs} \subset \mathcal{R}^{1mc}$. Since the boundary of each C-obstacle can be discretized into points, $\mathcal{R}^{1mc}_{obs}$ can be approximated by a list of points, which delimit the particular boundary $\mathcal{R}^{1mc}_{obs}$.

A point obstacle in the Workspace is $(x, y) \in \mathbb{R}^2$. Moreover, it maps into the Reachable Set of a single Motion Command, $\mathcal{R}^{1mc}$, as a region whose boundary can be approximated by points $(x_i, y_i) \in \mathbb{R}^2$. As such, the *obstacle information* in both spaces ($\mathcal{W}$ and $\mathcal{R}^{1mc}$) is fully represented by a set of points in $\mathbb{R}^2$. As a consequence, we can define a coordinate transformation over $\mathbb{R}^2$, that represents either $\mathcal{W}$ or $\mathcal{R}^{1mc}$.

In the following we will introduce the *Ego-Kinematic Transformation*, which maps $\mathbb{R}^2$ to the *Ego-Kinematic Space*, where the non-holonomic constraints are implicitly represented.

3 The Ego-Kinematic Space

The *Ego-Kinematic Space* (EK-space) is constructed by mapping points of $\mathbb{R}^2$ into: (1) descriptors of admissible paths leading to these points, and (2) the distances to reach these points measured over the admissible paths. This is made by means of the *Ego-Kinematic Transformation*. Since the kinematic constraints are embedded in the *Ego-Kinematic Transformation*, the admissible paths are mapped onto straight lines in the transformed space, and each point of the *EK-space* can be reached by a straight line motion "free-flying behavior".

To introduce the *Ego-Kinematic Transformation* (EKT), we focus our attention on those robots whose admissible paths were characterized by straight lines and arcs of a circle.

The EKT maps each point expressed in the robot's frame of reference into the *EK-space*, which we represent in polar coordinates for convenience. This mapping is a function of the circular arc length L and the radius R of the unique circular path leading to that point (see Fig. 2).

We then have:

$$\begin{array}{ccc} \text{EKT:}\mathbb{R}^2 & \rightarrow & \mathbb{R}^+ \times S^1 \\ (x, y) & \rightarrow & (d, \alpha) = (f_d(L, R), f_\alpha(R)) \end{array} \tag{4}$$

• The parameter d expresses the distance to a point $(x, y) \in \mathbb{R}^2$, measured along the circular path centered on the y-axis and containing (x, y) and the origin of the robot's reference frame. Let $R = \frac{x^2+y^2}{2y}$[1] be the radius of this circle. From the two possible values of arc length (L and $2\pi|R| - L$), the shortest distance is selected. This distance is calculated as:

[1] $R < 0$ represent circles whose center lies in $y < 0$.

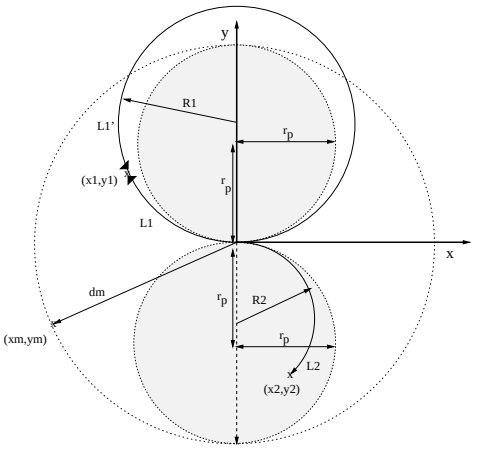

Figure 2: Circular paths leading to the points (x_1, y_1) and (x_2, y_2) of the space in the robot's frame of reference, see Fig 1.

$$f_d : \mathbb{R}^2 \rightarrow \mathbb{R}^+$$
$$(x,y) \rightarrow d = f_d(x,y) = \begin{cases} |R.acos(\frac{x^2-y^2}{x^2+y^2})|, & y \neq 0 \\ |x|, & y = 0 \end{cases}$$

Fig. 2 shows two points (x_1, y_1) and (x_2, y_2) in the robot's frame of reference and the arcs $d_1 = L_1$ and $d_2 = L_2$ calculated by this function.

• The angle α parameterizes the family of circular paths arising from the kinematic constraints. Instead of using $R \in [0, \infty[$ to parameterize the circles, we chose to use an angular descriptor in order to have a bounded representation.

We use the function $\alpha = \arctan(R/r_p)$, that converts into an index the comparison between a circular path with radius R, and one with *standard radius* r_p (where $d_m = 2.r_p$ is the diameter of the circular region to transform).

$$f_\alpha : \mathbb{R}^2 \rightarrow [-\pi, \pi]$$
$$(x,y) \rightarrow \alpha = f_\alpha(x,y) = \begin{cases} \frac{\pi}{2} - \arctan(\frac{|R|}{r_p}), & x \geq 0 \\ -\frac{\pi}{2} - \arctan(\frac{|R|}{r_p}), & x < 0 \end{cases}$$

A different value of α is used to distinguish the two possible paths over the same circle.

Figure 3a shows an example of the EKT applied to a discrete set of points of a polygon. Each side of the polygon is numbered to see how it transforms into the EK-space, illustrated in Figure 3b. The angles $\alpha \in]\alpha_1, \pi - \alpha_1[$ correspond to $0 < R < R_1$, whereas the angles $\alpha \in]\alpha_2, -\pi - \alpha_2[$ result from $0 > R > R_2$. The values of $\alpha = 0$ and $\alpha = \pi$ correspond to both,

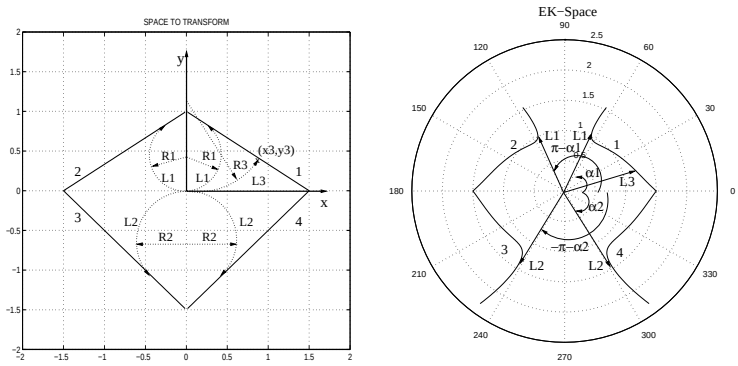

Figure 3: a) Space to be transformed. b) EK-space.

the straight forward and backward paths respectively along the x-axis.

The admissible paths to a point are represented as straight lines in the EK-space, as if the robot were free of kinematic constraints. Hence, the Euclidean distance in this space adequately describes the real distance from the robot to a point.

The Ego-Kinematic transformation is invertible. Once a pair $(d, \alpha) \in$ EK-space is selected, it determines unequivocally a pair $(x, y) \in \mathbb{R}^2$, and implicitly a $\theta \in S^1$, see Equation (3). It is worth to stress that when a value of α is fixed, a unique radius of turn is also fixed.

3.1 Forward Motion and Curvature Constraints

We next give guidelines for extending the *Ego-Kinematic Transformation* to be used for robots with additional motion and curvature constraints. First we consider the case of robots that can only perform forward motion. Then, we analyze the case of vehicles with curvature constraints, such as car-like robots.

The EKT transforms each point into parameters that unequivocally define a path towards that point. So far we have considered circular paths, where forward and backward motion are allowed. Here, we use the additional constraint of forbidding backward motion. This is easily added to the EKT by imposing a sense of direction when moving over a circle towards a given point. Fig. 2 illustrates this idea. To reach (x_1, y_1) we chose L_1', instead of L_1 that would be selected by the standard EKT.

Let us suppose now that the system has a minimum turning radius r_{min} (or a maximum curvature constraint γ_{max}), the case of a car-like robot. This can be easily incorporated in the EKT by the change of

variable $R' = R - r_{min}$, such that the space with $|R| < r_{min}$ is not transformed. (Since this region of the space cannot be reached in the execution of a single motion command, there is no need to transform it).

4 Using the *EK-space* for Reactive Navigation

We finally address the generation of collision-free motion for non-holonomic robots by using reactive navigation methods.

Without taking the robot kinematic constraints into account, the execution of the motion generated by a reactive method is doomed to rely on some approximations. Instead, we use the EK-space to represent the robot kinematic constraints. Then, reactive methods, which are not designed for non-holonomic robots, can be applied over the EK-space with full generality, irrespective whether the method is formulated in the Workspace or Configuration space. Finally, the reactive method solution is transformed back to the Workspace to calculate the motion command for the robot.

At each sampling time T, the procedure to use the EK-space is the following:

1. The obstacle information is reduced to points expressed in the robot's frame of reference. If the reactive method is defined over the Workspace $\mathcal{W}$, the EKT is directly applied to the obstacle points. Otherwise, if the reactive method applies to the Configuration space, we first have to build the C obstacle region $\mathcal{R}_{obs}^{1mc}$ as described in Section 2, to use the EKT. In both cases the result is the obstacle information expressed in the EK-space.

2. The reactive method is then applied in the EK-space to compute the desired motion direction α.

3. A turning radius is calculated by $R = EKT^{-1}(\alpha)$. Finally, a motion command that preserves this turning radius is obtained, having $v = \omega R$.

To demonstrate this methodology we have conducted some experiments using two very different reactive navigation methods, whose original formulation does not take the kinematic constraints into account. Firstly, we discuss the Nearness Diagram Navigation [12], applied to the workspace and which allows forward motion only. Secondly, we use a Potential Field Method [1], defined in the Configuration space and allowing both forward and backward motion.

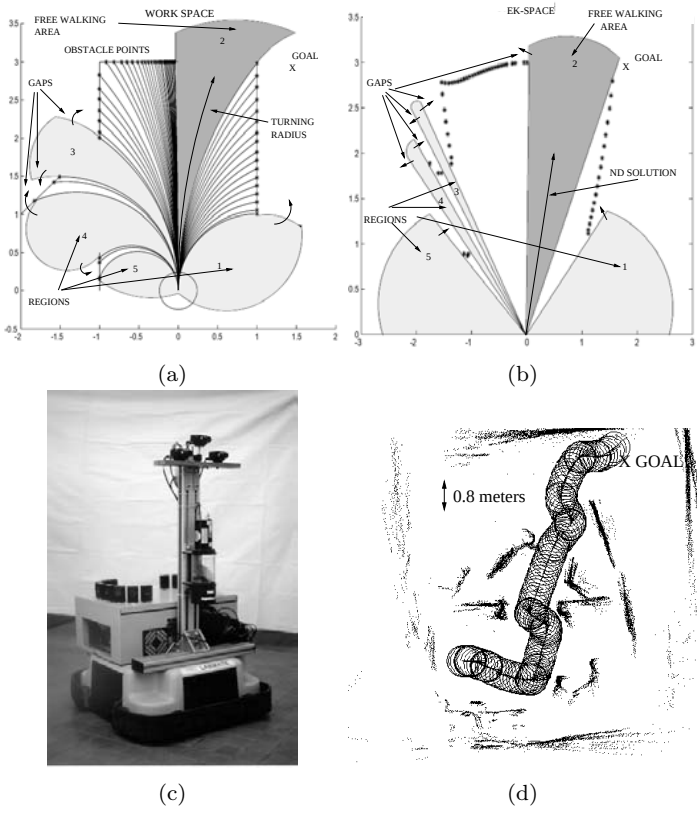

Figure 4: a)Workspace . b)EK-space. c) Labmate robot. d) Experiment.

We remark that the results obtained are not bench marked with other methods. This is motivated by the fact that the scope of the paper is the spatial representation, not the reactive method itself. The particular results are completely dependent on the reactive method used and the specific implementation.

4.1 Nearness Diagram Navigation

The *Nearness Diagram Navigation (ND)* [12] is a reactive navigation method that does not take into account the kinematic constraints in its formulation.

The ND method is based on the situated-activity methodology of design [13]. First, a set of five situations that fully describe the relative state of the robot, obstacle distribution, and goal location are defined. Subsequently, one action is designed for each situation. In real-time the perception is used to identify the current situation, and the associated action is executed, which generates the motion commands.

To identify the current situation, some high level infor-

mation of the environment is searched: gaps, regions, free walking areas. And some criteria are also applied: a security criterion, a free walking area criterion, etc. The individual action designs are based on a geometrical implementation.

We next use the EK-space to apply the ND method to robots with kinematic constraints.

At each sample period the following procedure is repeated:

1. The obstacle information in the Workspace (Fig. 4a) is transformed into the EK-space (Fig. 4b).
2. The ND method is applied in the EK-space to calculate the most suitable motion direction α^* (represented in Fig. 4b as the ND solution). In Fig. 4b we also show some information used by the ND to calculate the solution (the gaps, regions, and the free walking area are also shown in Fig. 4a).
3. This direction α^* is transformed back to the Workspace as a turning radius $R^* = EKT^{-1}(\alpha^*)$, see Fig. 4a. Finally, a motion command that preserves this turning radius is calculated.

We have tested our approach in a Labmate robot, Fig 4c, to generate reactive collision-free motion. The platform is a two-driving wheels robot with a square geometry that we approximate by a circle due to ND implementation requeriments. The main sensor is a 3D laser. Fig 4d shows an experiment where the robot was safely driven towards the goal location - the only information given a priori to the robot. The execution time of the algorithm (construction of the EK-space and reactive method usage) is around $150msec$ on a $100MHz$ microSun SparcII, which was adequate for real time.

4.2 Potential Field Method

Potential Field Methods (PFM) [1] have been widely used by many researchers, in spite of the fact that they cannot be applied to non-holonomic robots.

The robot is considered as a particle in the Configuration space, moving under the influence of an artificial potential field. While the goal creates a potential that attracts the particle, the obstacle information creates a repulsive one. The motion is generated to follow the direction of the artificial force induced by the sum of the two potentials.

This framework cannot be used with non-holonomic robots, because the gradient direction of the potential does not preserve the non-holonomic constraint of Equation (1). In other words, the potential structure does not represent that not all differential motions are allowed in the Configuration space.

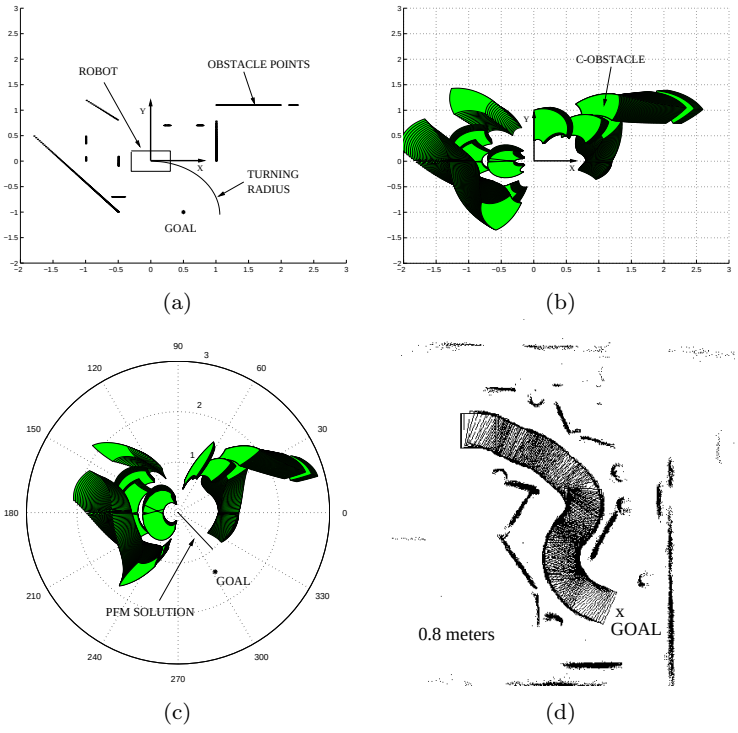

Figure 5: (a) Workspace. (b) Reachable Obstacle Set of a single motion command $\mathcal{R}_{obs}^{1mc}$ induced by the robot's admissible paths and geometry. (c) EK-space. (d) Experimental result.

Using the EK-space, we can apply the PFM to non-holonomic robots, repeating the following procedure at each sampling instant:

1. The obstacle points in the workspace, Fig. 5a, are used to construct the C-obstacle region $\mathcal{R}_{obs}^{1mc}$ in the Reachable Set of a single Motion Command, see Fig. 5b. Then, the EKT is applied to $\mathcal{R}_{obs}^{1mc}$, yielding the obstacle information in the EK-space, which is illustrated in Fig. 5c.
2. The Potential Field Method is applied to the EK-space to calculate the most promising direction of motion α^*, represented in Fig. 5c as the PFM solution.
3. This direction α^* is transformed back to the workspace as a turning radius (see Fig. 5a), determined by $R^* = EKT^{-1}(\alpha^*)$. Finally a motion command that preserves this turning radius is calculated.

We also tested this procedure in the Labmate platform. In this case the robot geometry is directly taken into account due to the PFM formulation. We have used the potential field functions proposed in [14]. Fig. 5d shows an experiment performed in the real platform, showing how this method was able to drive the constrained platform free of collisions to the goal location - the only information given a priori to the robot. The execution time of the algorithm (construction of the Reachable Set of a single Motion Command, EK-space and reactive method usage) is about $250msec$, hence was well suite for real-time.

5 Conclusions

In this paper we have addressed the problem of applying reactive navigation methods to non-holonomic robots.

Our approach to this problem is based on the idea of using a representation of the space, that is consistent with the motion constraints of a given vehicle. We call this spatial representation the *Ego-Kinematic Space*. We have defined the *Ego-Kinematic Transformation* that maps the Euclidean space to the *Ego Kinematic Space*, by taking the non-holonomic constraints of the vehicle into account.

The advantage of using the *Ego-Kinematic Space* is that reactive navigation methods can be directly applied to this space, and as a consequence, verify the vehicle kinematic constraints without having to address them explicitly. This methodology provides a general solution to apply reactive navigation methods to non-holonomic systems.

The *Ego-Kinematic Space* construction does not impose a critical time penalty and it can be used in real time. The *Ego-Kinematic Transformation* is a bi-dimensional transformation, and the complexity of the algorithm increase linearly with the number of obstacle points considered to transform.

We have shown how the *Ego-Kinematic Transformation* can be used for many types of mobile platforms. Moreover, the *Ego-Kinematic transformation* can map either the Workspace or to the so-called *Reachable Set of a single Motion Command*. Thus, it can be used with a large number of reactive methods. We have shown experiments with two reactive navigation methods that do not take into account the kinematic constraints. Making use of the EK-space, these methods succeeded in driving a kinematicaly and geometrically constrained platform towards a given location, while avoiding collisions.

6 Acknowledgements

The research described in this paper was partially supported by the Spanish CICYT project DPI2000-1272 ad by the EU TMR network SMART-II.

References

[1] O.Khatib "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots" *The International Journal of Robotics Research*, 5(1), 1986.

[2] J.Borenstein and Y.Koren "The Vector Field Histogram (VFH)-Fast Obstacle Avoidance for Mobile Robots" *IEEE Transactions on Robotics and Automation*, 7(3), 1991.

[3] I.Ulrich and J.Borenstein "VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots" *IEEE International Conference on Robotics and Automation*, pp1572, Leuven, Belgium, 1998.

[4] W.Feiten R.Bauer G.Lawitzky "Robust Obstacle Avoidance in Unknown and Cramped Environments" *IEEE International Conference on Robotics and Automation*, pp2412, 1996.

[5] R.Simmons "The Curvature-Velocity Method for Local Obstacle Avoidance" *IEEE International Conference on Robotics and Automation*, pp3375, Minneapolis, USA, 1996.

[6] D.Fox W.Burgard S.Thrun "The Dynamic Window Approach to Collision Avoidance" *IEEE Transactions on Robotics and Automation*, 4:1, 1997.

[7] Sean Quinlan "Real-Time Modification of Collision Free Paths" *PHD thesis, Stanford University* December, 1994.

[8] Maher Khatib "Sensor-based motion control for mobile robots" *PHD thesis, LAAS-CNRS* December, 1996.

[9] J.C.Latombe "Robot Motion Planning" *Kluwer Academic Publishers*, 1991.

[10] J.P.Laumond S.Sekhavat F.Lamiraux "Guidelines in Nonholonomic Motion Planning for Mobile Robots" in *Robot Motion Planning and Control*, J.P.Laumond, Ed New York: Springer-Verlag, 1998, vol. 229.

[11] Oliver Brock "Generation of Robot Motion: Integrating Planning and Execution" *PHD thesis, Stanford University* 1999.

[12] J.Minguez L.Montano "Nearness Diagram Navigation: A New Real Time Collision Avoidance Approach" *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)*, Takamatsu, Japon, 2000.

[13] R.Arkin "Behavior-based robotics" *MIT Press* 1998.

[14] L.Chengqing M.H.Ang L.S.Yong "Virtual obstacle concept for local-minimum-recovery in potential-field based navigation" *IEEE International Conference on Robotics and Automation* pp. 983-988. San Francisco, USA, 2000.