Proceedings of the 2002 IEEE/RSJ
Intl. Conference on Intelligent Robots and Systems
EPFL, Lausanne, Switzerland • October 2002

Reactive Collision Avoidance for Navigation with Dynamic Constraints

Javier Minguez[1], Luis Montano[1], Oussama Khatib[2]

[1] *Dept. of Computer Science and Systems Engineering, University of Zaragoza (Spain)*[*]
[2] *Robotics Laboratory, Stanford University (USA)*

Abstract

We address the problem of applying reactive navigation methods for collision avoidance to systems where the dynamics cannot be neglected: mobile robots with slow dynamic capabilities, or systems working at high speeds. Rather than embedding the motion constraints when designing a navigation method, we propose to introduce the robot dynamic constraints directly into the spatial representation. In this space the dynamic capabilities of the robot are implicitly represented. With minor modifications, standard reactive navigation methods can be used in this space implicitly taking into account the robot dynamic constraints. To validate this framework, we show experimental results using two reactive navigation methods whose original formulation do not take the robot dynamic constraints into account (the Nearness Diagram Navigation and the Potential Field method).

1 Introduction

This paper addresses the problem of reactive collision avoidance for systems where the dynamics cannot be ignored. Even thought the majority of robots exhibit dynamic constraints, most of the reactive navigation methods do not take the dynamic constraints into account. Then, these methods are susceptible to failure when the vehicle dynamics take an important role: (1) Systems working at high speeds, or (2) systems with slow dynamics. Examples of these methods include: Potential Field methods [1], Vector Field Histogram [2], [3], Elastic Band [4], Elastic Strips [5], Nearness Diagram Navigation [6].

The dynamic constraints have been mainly addressed in sensor-based motion planning from two different points of view: (1) some researches deal with the problem of the dynamics by modeling the system behavior. Some of them directly model the system [7], [8], [9]. Others identify the system model by the responses to motion commands (inputs) [10], [11]. Once the model is available, the system responses are also known and are used to apply reactive navigation strategies. (2) Some authors explain the system response with a model of constrained inputs. Some of them translate the reactive navigation problem to the motion command space, and solve it as a constrained optimization [12], [13], [14]. Others calculate dynamic admissible trajectories to obtain the motion commands later on [15], [16].

The main contribution of this work is a solution to incorporate dynamics into reactive collision avoidance methods. We propose to use the dynamic constraints to build a new spatial representation - *Ego-Dynamic Space* - where the dynamic constraints are implicitly represented. Then, with minor modifications, off-the-shelf reactive navigation methods that do not take the dynamic constraints into account can be applied to this space. The motion commands calculated implicitly take the specific robot dynamics into account assuring feasible motion execution.

To demonstrate and validate the usefulness of this framework, we have extended and experimentally tested two reactive collision avoidance approaches that do not address the dynamic constraints into their formulation - the Potential Field method [1] and the Nearness Diagram Navigation [6].

2 Preliminaries on Reactive Navigation

To achieve the goals of this paper, we turn to a discussion about the vehicle case of study, the reactive methods, the motion commands, and the role of the dynamic constraints.

2.1 Vehicle case of study

We focus our attention on a circular and holonomic robot. The workspace $\mathcal{W}$ is $\mathbb{R}^2$ and the configuration space for this robot $\mathcal{C}$ is $\mathbb{R}^2$ (ignoring the robot orientation). Let be $\mathbf{v} = (v_x, v_y)$ the motion command (expressed in the robot's reference system).

[*]The research described in this paper was partially supported by the Spanish CICYT project DPI2000-1272.

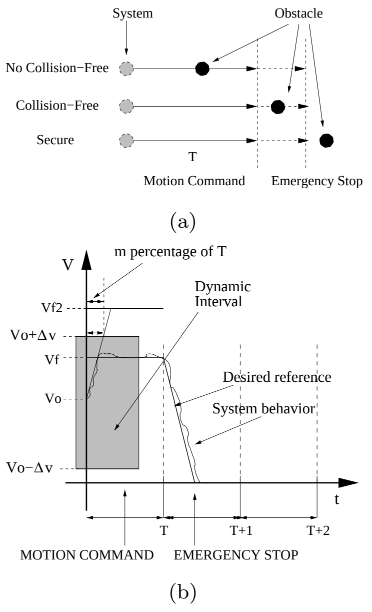


Figure 1: *a) Types of motion commands. b) Motion command execution.*

2.2 The Reactive Navigation Problem

The reactive navigation methods compute at each sample period a collision-free motion command that drives the robot towards a goal location. These techniques have been demonstrated to work well in unknown, dynamic, and non-predictable environments.

The spaces where usually the reactive navigation methods apply are the workspace $\mathcal{W}$, e.g. [6], [5], [2]; or the configuration space $\mathcal{C}$ [17], e.g. [1], [8], [3], [4]. The research presented here is based on a spatial representation - prior to the reactive method use - where the system dynamics are implicitly represented. We then analyze both spaces, $\mathcal{W}$ and $\mathcal{C}$, to achieve the maximum generality.

2.3 Motion Commands

Based on a perception-action process, the reactive navigation methods calculate at each instant the *"best" motion command*: To avoid collisions whilst moving the robot towards a given goal location. Let us classify the types of motion commands as follows:

- *Emergency Stop*: this command is a policy to stop the robot applying the maximum deceleration of the system.
- *Collision-Free commands*: the execution of these motion commands is free of collisions during the next sample time T.
- *Secure commands*: these motion commands assure: (1) the execution is collision-free during the next sample time T - they are *Collision-Free*, and (2) after the execution of the motion command, the robot can be stopped with the *Emergency Stop* without collide.

Fig. 1a illustrates the motion commands in the one-dimensional case. The system executes a motion command and later it is stopped by an *Emergency Stop.* The first motion command is not *Collision-Free.* The second motion command is *Collision-Free*, but let us stress that after the command execution, the robot cannot avoid the collision. The last motion command is a *Secure command.* This command produces a *Collision-Free* motion during T, and later, the command assures that the robot can be stopped safely if it is required with an *Emergency Stop.* In reactive navigation it is desirable to generate this type of motion commands: *Secure commands.*

2.4 The Dynamic Constraints

Let us introduce the two dynamic constraints that the maximum system acceleration-deceleration, a_b, establishes in the execution of a motion command:

1. *Braking constraint*: When the *Emergency Stop* is launched, the controller is designed to apply the maximum deceleration, a_b, over time to stop the robot, see Fig 1b.

2. *Dynamic interval*: The controller is designed to reach as soon as possible the steady state of a reference command, v_f. First, the maximum acceleration of the system, a_b, is applied to reach the reference velocity. Subsequently the steady state is maintained, see Fig. 1b. Given the current robot velocity, v_o, a new command is *dynamic admissible* if it is within the dynamic interval $v_{next} \in [v_o \pm \triangle v]$. Let us assume that the system can instantaneously achieve the desired velocity - the robot moves at a constant velocity during the sampling period T. Let us fix m (m is the sampling period T percentage for the system to reach the steady state), then $\triangle v = |a_b|.m.T$ is fixed. The position error between the real system behavior and the constant velocity assumption is $e_d = \frac{|a_b|.(m.T)^2}{2}$. If e_d is out of our requirements, a lower m is fixed and $\triangle v$ is reduced.

The first constraint fixes the maximum distance that the robot travels when the *Emergency Stop* is launched. The second constraint establishes the set of dynamic feasible motion commands that can be selected. We next present the design of the spatial transformation to embed the deceleration capabilities of the system into the spatial representation.

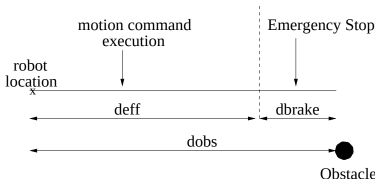


Figure 2: *a) Distance to an obstacle.*

3 The Ego-Dynamic Space

The idea is to build a spatial representation where the distances to the obstacles are transformed into distances that depend on the robot deceleration constraint, and on the sampling time. In this space - *Ego-Dynamic Space* - the first dynamic constraint presented in Subsection 2.4 will be represented.

Let us start by studying the problem in one-dimension, see Fig. 2a. Let be d_{obs} the real (measured) distance from the robot to an obstacle. Let be d_{eff} the *effective distance*: the maximum distance that the robot can travel at a constant velocity during the period T, allowing later the *Emergency stop* (applying the maximum deceleration a_b) for stopping the robot safely before hitting the obstacle. Then:

$$d_{obs} = d_{eff} + d_{brake}$$
$$d_{eff} = v.T \qquad d_{brake} = \frac{v^2}{2.a_b}$$
$$\frac{d_{eff}^2}{2.a_b.T^2} + d_{eff} - d_{obs} = 0 \tag{1}$$

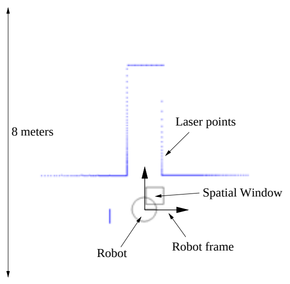
where we obtain d_{eff}, and thus the *Ego-Dynamic Transformation* (EDT):

$$\begin{array}{lcl} \text{EDT:}\mathbb{R}^+ & \rightarrow & \mathbb{R}^+ \\ d_{obs} & \rightarrow & d_{eff} = a_b.T^2.(\sqrt{1 + \frac{2.d_{obs}}{a_b.T^2}} - 1) \end{array} \tag{2}$$

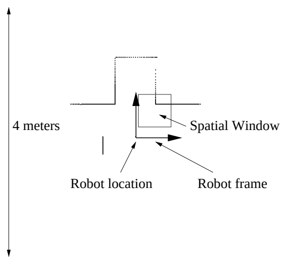
This distance, d_{eff}, is a motion constraint: if the robot moves farther than d_{eff}, the robot could not be safely stopped with the *Emergency Stop*. d_{eff} depends on: (1) the measured distance to the obstacle, d_{obs}. (2) The deceleration capabilities of the robot, a_b. (3) The sampling time, T, in which the next motion command will be applied.

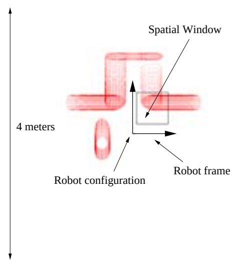
In reactive navigation sometimes the deceleration effects are ignored, it is assumed infinite deceleration capabilities. This is fully represented by Equation (1): when $a_b \rightarrow \infty$, d_{eff} tends to be the real (measured) distance d_{obs} used by these methods.

This framework extends for two dimensions with generality. As the braking trajectory is not a straight line (parabola) we identify the error with respect to the one-dimensional case. Deeper details are out of the scope of the paper, but let us remark that the error can be depreciated in the context of reactive navigation: obstacle distances $d_{obs} < 3m$, sample periods $T < 0.5sec$, and system accelerations $a_b < 1.0\frac{m}{sec^2}$. In this case the upper bound is around $0.1m$.



(a)





(b) (c)

Figure 3: *a) Workspace. b) ED-space from the workspace. c) ED-space from the configuration space.*

We will refer to this spatial representation as the *Ego-Dynamic space* (ED-space). The *Ego-Dynamic Transformation* can be then applied to both, the workspace and the configuration space, laying the obstacle information in the ED-space. Let us illustrate the advantage of the ED-space by an example:

Example 1 *Fig. 3 shows an example of the* EDT *applied to both, the $\mathcal{W}$ and $\mathcal{C}$. Fig. 3a illustrates the robot in the workspace and the current perception (laser scan). We fix $a_b = 1\frac{m}{sec^2}$ and $T = 0.5sec$. The EDT is directly applied to the obstacle points in $\mathcal{W}$ yielding the obstacle information in the ED-space (see Fig. 3b), that is expressed in the robot's frame of reference. To transform $\mathcal{C}$, we first build the $\mathcal{C} - Obstacles$ by enlarging the obstacle points with the robot radius. The result of applying the EDT is the obstacle information in the ED-space, Fig. 3c. Comparing the workspace and the ED-space, we observe that the obstacle information is closer to the robot when the dynamic constraints are considered. The ED-space fully represents the dynamic capability: the obstacles must be taken into account before than ignoring the dynamics - that is, the obstacles are represented closer to the robot.*

The main advantage of this spatial representation is that the robot deceleration and sample period are implicitly represented in the space. The next Section presents the *Spatial Window* to address the second dynamic constraint: the dynamic interval.

4 The Spatial Window

This Section presents the concept of *Spatial Window* to deal with the second dynamic constraint presented in Subsection 2.4: the next motion command has to be within the dynamic interval $[\mathbf{v_o} \pm \triangle \mathbf{v}]$.

The *Spatial Window* (SW) is the set of possible robot locations that can be attained by motion commands within the admissible dynamic interval. Assuming a constant velocity during the period T, the corners of the *Spatial Window* are given by:

$$X_{max} = (v_{ox} + \triangle v).T \quad \text{and} \quad X_{min} = (v_{ox} - \triangle v).T$$
$$Y_{max} = (v_{oy} + \triangle v).T \quad \text{and} \quad Y_{min} = (v_{oy} - \triangle v).T$$

where $\mathbf{v_o} = [v_{ox}, v_{oy}]$ is the current velocity.

Figs. 4a,b depict an example of the SW in $\mathcal{W}$ and $\mathcal{C}$. In the workspace $\mathcal{W}$, Fig. 4a, each obstacle point creates an *obstacle line* of locations that cannot be attained with the execution of a single motion command without collisions. The intersection of these obstacle lines with the SW gives the collision locations. In the configuration space $\mathcal{C}$, Fig. 4b, the obstacle points create the $\mathcal{C}-$*Obstacles*. The procedure is the same, but each $\mathcal{C}-$*Obstacle* produces an *obstacle region* instead of an *obstacle line*, see Fig. 4b. In both cases, the SW holds the locations or configurations that can or cannot be attained without collisions, with the execution of a single motion command within the dynamic interval.

Once a collision-free position $\mathbf{x_p} = (x_p, y_p)$ within the *Spatial Window* is calculated, a *Collision-Free command* is given by $\mathbf{v} = (\frac{x_p}{T}, \frac{y_p}{T})$. The main interest of the *Spatial Window* is that the motion command $\mathbf{v}$ is *dynamic feasible* by the system because is within the admissible dynamic interval, assuring feasible motion **execution**.

The objective of this work is to fully take into account the dynamics into the motion generation layer. Moreover, the motion commands calculated have to be *Secure commands* as presented in the previous Sections. In the next Section we show how to combine both the *Ego-Dynamic Space* and the *Spatial Window* to fulfill both requirements.

5 Combining the SW and the ED-space

The goal of this Section is to unify in a framework the *Ego-Dynamic Space* and *Spatial Window* to: (1) calculate *Secure commands*, see Subsection 2.3, to assure robust motion commands **calculation**. (2) To fully take into account the dynamics involved in the motion command generation, see Subsection 2.4, to assure feasible motion commands **execution**.

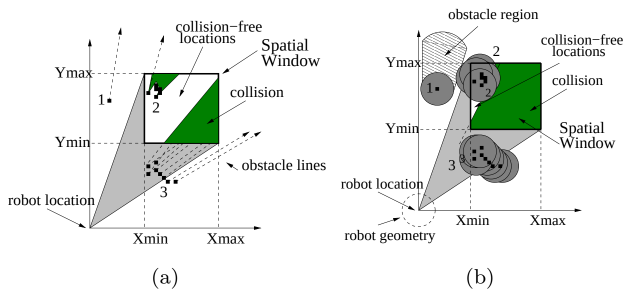


Figure 4: *a) Spatial Window in the workspace. b) Spatial Window in the configuration space.*

At each sampling time T the procedure is:

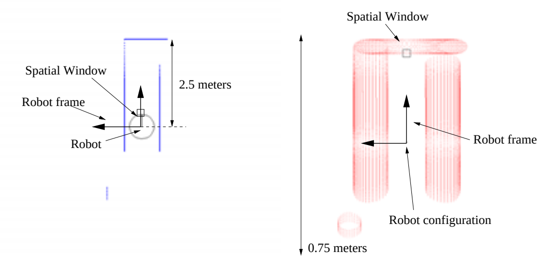
1. The obstacle information is reduced to points expressed in the robot frame of reference. To transform the workspace, $\mathcal{W}$, the EDT is directly applied to the obstacle points. Otherwise, to transform the configuration space, $\mathcal{C}$, we first build the $\mathcal{C}-$*Obstacle* region, to apply the EDT. In both cases the result is the obstacle information expressed in the ED-space.

2. The SW is applied in the ED-space. The locations that cannot be attained due to the obstacle distribution are labeled as not-collision-free in the SW, following the procedure presented in Section 4.

3. Any strategy to select one collision-free location, $\mathbf{x_p}$, within the SW is valid. Then a motion command is directly calculated by $\mathbf{v} = (\frac{x_p}{T}, \frac{y_p}{T})$. In the next Section we will present how to use reactive navigation methods to achieve this goal.

We next show two examples that highlight the relevance of this framework:

Example 2 *Figs. 3a,b,c show this framework with a system working at high speeds with a fast dynamic capability (maximum acceleration $a_b = 1\frac{m}{sec^2}$). The sample period is $T = 0.5sec$. Fig. 3a shows the robot in the workspace moving with $\mathbf{v_o} = (0.6\frac{m}{sec}, 0.8\frac{m}{sec})$ that fixes the SW in the space. The SW is free of obstacles and all locations give* Collision-Free *motion commands for the next sample period. On the other hand, if we apply the EDT to $\mathcal{W}$ or $\mathcal{C}$, we obtain the obstacle information in the ED-space, see Figs. 3b,c. In both cases the ED-space represent the obstacles closer to the robot due to the dynamic capabilities, and thus there are obstacles within the SW that constraint the possible locations that can be chosen. Once a collision-free location within the SW in the ED-space is selected, a* Secure command *is given. Let us stress that without taking the dynamics in this situation the robot might crash the corner.*



(a) (b)

Figure 5: *a)Workspace. b)ED-space from $\mathcal{C}$.*

Example 3 *An example with a system with extreme slow dynamics, $a_b = 0.1\frac{m}{sec^2}$, is illustrated in Fig. 5. The robot moves in the environment presented in Fig. 3a located in the center of the corridor, facing the frontal wall, and moving at $\mathbf{v_o} = (0.61\frac{m}{sec}, 0.0\frac{m}{sec})$ towards the frontal wall (2.5m), see Fig 5a. In the resulting ED-space (see Fig 5b) the obstacles are very close to the robot location - due to the slow dynamics. Moreover there are some obstacles within the SW. The particular collision-free locations in the SW are those closer to the robot. These locations will produce motion commands that reduce the velocity of the robot. This is the correct behavior: the robot starts to stop very early to avoid the collision due to the limited dynamic capability - even if the obstacle is located far away, 2.5m. Without taking the dynamics in this case, the robot will crash the frontal wall.*

In this framework it is possible that all locations would not be collision-free in the SW. Then, the *Emergency Stop* is launched to safely stop the robot. Then the motion is resumed.

The main interests of the framework are:

1. The motion commands calculated are feasible for the specific system dynamics. They are within the dynamic interval $\mathbf{v_{next}} = [\mathbf{v_o} \pm \triangle \mathbf{v}]$. This follows from the fact of using the SW.

2. The motion commands are *Secure commands*, because they are calculated from locations in the ED-space. The motion commands are *Collision-Free* in execution during the sampling time T, while giving the guaranty for stopping the robot safely in the next time if it is required.

The next Section presents the strategy that we use - the reactive navigation methods - to calculate the desired location within the SW, that implicitly fix the motion command.

6 Reactive Navigation with Dynamic Constraints

The framework presented in the previous Section opens the problem of selecting one location within the SW, that implicitly fixes a *Secure command.* This Section presents a solution based on reactive methods to select locations within the SW.

Let us stress that any strategy to choose one collision-free location within the SW solves the problem. [13], [14], and [12] solve a similar problem by a constrained optimization that balances the goal location, the forward progress, and the obstacle clearance. Similar strategies could be used.

Our solution is based on the use of off-the-self reactive navigation methods in the ED-space to select a collision-free location within the SW. The main motivations to choose these techniques are:

1. The presented framework has to work in real-time. The use of a reactive navigation method does not impose a significant time penalty, and both -*ED-space + SW + Reactive method* - can be run in real time.

2. The reactive navigation methods take directly into account the goal location; and information of the obstacle spatial distribution - this advantage is lost when the problem is solved with other type of heuristics, e.g. with a constrained optimization as [13], [14], [12].

3. The reactive navigation method is a module completely independent of the rest of the framework. Different reactive methods can be used in the framework without significant changes.

Most of the reactive navigation methods give as solution a *most promising motion direction* [1], [8], [2], [3], [4], [5], [6], among others. The intention is to apply the reactive navigation method into the ED-space, and use the direction solution to compute a collision-free location within the SW. The location is selected as follows:

1. The reactive method direction solution intersects the SW. The closest location (within the SW) to the direction solution, and farthest to the robot location is selected. This heuristic privileges the maximum forward progress, while moving the robot towards the closest location to the reactive method solution.

2. The reactive method direction solution does not intersect the SW. The closest location (within SW) to both, the robot location and the direction solution is selected. This heuristic reduces

the robot velocity while moving the robot towards the reactive method solution.

7 Experimental results

This Section validates experimentally the presented research. For experimentation, we used a *Nomadic XR4000* robot. This robot is a circular holonomic platform. We estimated that the physical acceleration-deceleration of the platform that is around $0.75 \frac{m}{sec^2}$. The sensor used is a 2D SICK laser.

We present experiments conducted with two reactive navigation methods: the Nearness Diagram Navigation [6], that applies to the workspace $\mathcal{W}$. Secondly, we use the Potential Field Method [1], that is used in the configuration space $\mathcal{C}$. Both methods do not take into account the dynamic constraints. Then, the computed motion commands do not correspond to feasible motions. The robot executes paths that are not the desired ones: collision avoidance cannot be longer guarantied. By using the *ED-space + SW + Reactive method*, the computed motion commands are feasible for the robot. The robot can execute the computed commands. We focus on this issue in this Section.

Figure 6: *The Nomad XR4000*

In all the experiments the environment was unknown, unstructured, and could be dynamic. Only the goal location was available in advance. Under these circumstances the use of reactive navigation methods to move the robot is justified.

7.1 Nearness Diagram Navigation

The *Nearness Diagram Navigation (ND)* [6] is a reactive navigation method that does not take into account the dynamic constraints in its formulation.

We tested the framework with the ND on the real platform. Fig. 7a shows the result of one of our runs. The robot successfully reached the goal location while it avoided collisions with the environment. Fig. 7b,c depict the behavior of the system during the experiment: the motion commands reference calculated by the framework (v_x and v_y), and the real ones executed by the robot (for better appreciation only a fraction of the experiment is shown).

7.2 Potential Field Method

The Potential Field Methods (PFM) [1] can be used as reactive navigation methods, but they do not take the dynamic constraints into account.

Fig. 7d shows the result of one of our runs with the framework and a PFM. Fig. 7e,f depict the motion commands reference calculated by the framework (v_x and v_y), and the real ones executed by the robot (only a fraction of the experiment is shown).

The velocity profiles of both experiments illustrate how the computed motion commands are feasible by the platform. Thus the robot executes paths that closely match with the computed ones. Moreover, the computed motion commands are *Secure commands*: (1) they are *Collision-Free* during the execution time, and (2) they give the guarantee for stopping the robot safely subsequently with an *Emergency Stop* policy. We achieve these goals by addressing the system dynamics.

8 Conclusions

We address the problem of applying reactive navigation methods to systems where the dynamic constraints cannot be neglected. We have presented a general framework to take into account the dynamics of the systems into the reactive navigation layer. Moreover, we have presented experimentation with two reactive methods that originally do not take into account dynamics. The *Nearness Diagram Navigation* and the *Potential Field methods*

The research presented here assumes that the reactive motion is generated for a circular and holonomic robot. The future work will follow the direction of extending the presented research for systems with non-circular shapes and with kinematic constraints.

9 Acknowledgments

We wish to thank all the members of O. Khatib's research group who hosted J. Minguez's visit to Stanford University. In particular, we thank O.Brock and J.Park for their valuable comments and discussions. Moreover we thank D.Ortín of University of Zaragoza (Spain) for the valuable comments and discussions.

References

[1] O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," *Int. Journal of Robotics Research*, vol. 5, pp. 90–98, 1986.

[2] J. Borenstein and Y. Koren, "The Vector Field Histogram–Fast Obstacle Avoidance for Mobile Robots," *IEEE Transactions on Robotics and Automation*, vol. 7, pp. 278–288, 1991.

[3] I. Ulrich and J. Borenstein, "VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots," in *IEEE Int. Conf. on Robotics and Automation*, 1998, pp. 1572–1577.

[4] S. Quinlan and O. Khatib, "Elastic Bands: Connecting Path Planning and Control," in *IEEE Int.*

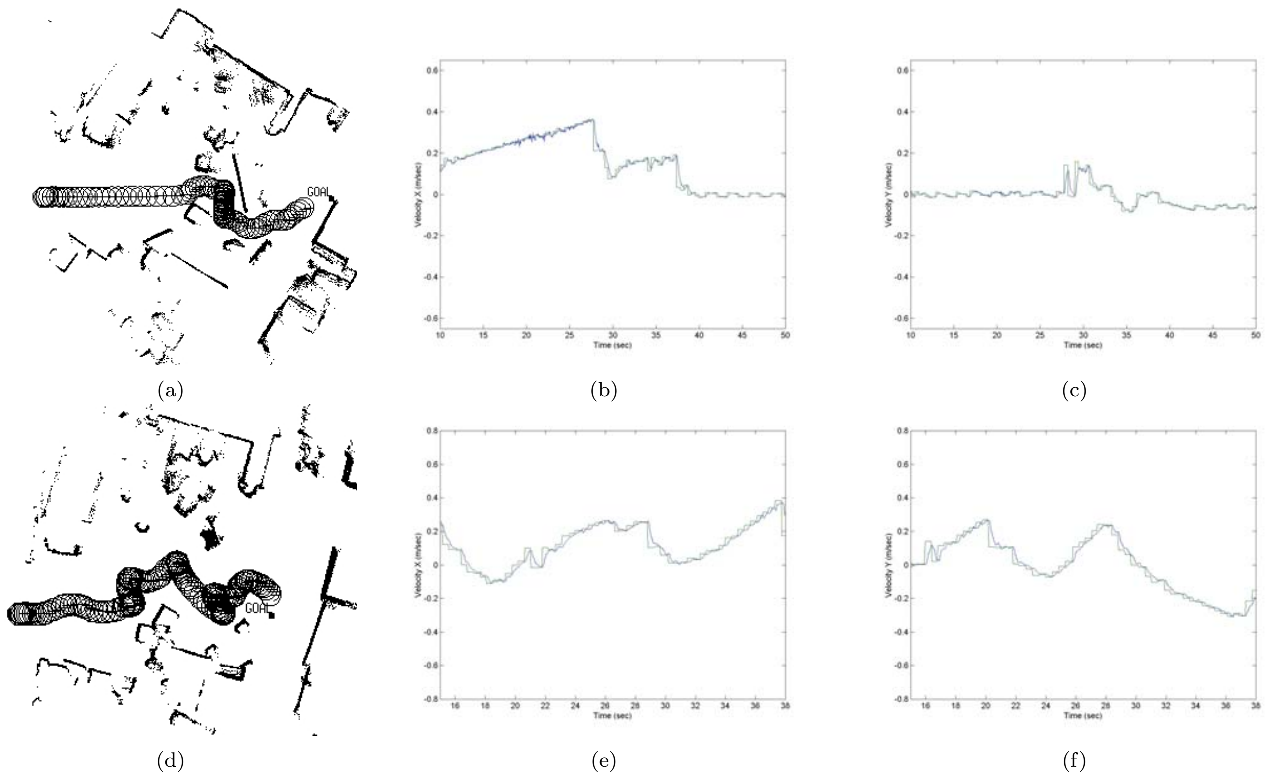


Figure 7: *a,b,c) Experiment with ND and velocity profiles. d,e,f) Experiment with PFM and velocity profiles. The velocity profiles depict the computed and real robot velocities.*

Conf. on Robotics and Automation, Atlanta, USA, 1993, vol. 2, pp. 802–807.

[5] O. Brock and O. Khatib, "Real-Time Replanning in High-Dimensional Configuration Spaces using Sets of Homotopic Paths," in *IEEE Int. Conf. on Robotics and Automation*, San Francisco, USA, 2000, pp. 2328–2334.

[6] J. Minguez and L. Montano, "Nearness Diagram Navigation (ND): A New Real-Time Collision Avoidance Approach," in *IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, Takamatsu, Japan, 2000, pp. 2094–2100.

[7] O. Khatib and et al, "Robotics and interactive simulation," *Communications of the ACM*, 2002.

[8] J. Asensio and L. Montano, "A Kinematic and Dynamic Model-Based Motion Controller for Mobile Robots," in *15th IFAC World Congress*, Barcelona, Spain, 2002.

[9] A. Calcitti and R. Zapata, "Reactive Behaviours of Mobile Manipulators Based on the DWZ Approach," in *IEEE Int. Conf. on Robotics and Automation*, Korea, 2001.

[10] J.C. Alvarez, A. Shkel, and V. Lumelsky, "Building topological models for navigation in large scale environments," in *IEEE International Conference on Robotics and Automation*, Leuven, Belgium, 1998.

[11] V. Lumelsky and A. Shkel, "Incorporating body dynamics into the sensor-based motion planning paradigm. the maximum turn strategy," in *IEEE International Conference on Robotics and Automation*, Nagoya, Japan, 1995.

[12] R. Simmons, "The Curvature-Velocity Method for Local Obstacle Avoidance," in *IEEE Int. Conf. on Robotics and Automation*, Minneapolis, USA, 1996, pp. 3375–3382.

[13] D. Fox, W. Burgard, and S. Thrun, "The Dynamic Window Approach to Collision Avoidance," *IEEE Transactions on Robotics and Automation*, vol. 4, no. 1, 1997.

[14] O. Brock and O. Khatib, "High-Speed Navigation Using the Global Dynamic Window Approach," in *IEEE Int. Conf. on Robotics and Automation*, Detroit, MI, 1999, pp. 341–346.

[15] W. Feiten, R. Bauer, and G. Lawitzky, "Robust Obstacle Avoidance in Unknown and Cramped Environments," in *IEEE Int. Conf. on Robotics and Automation*, 1996, pp. 2412–2418.

[16] T. Wilkman, M. Branicky, and W. Newman, "Reflexive Collision Avoidance: A Generalized Approach," in *IEEE Int. Conf. on Robotics and Automation*, 1993.

[17] J. C. Latombe, *Robot Motion Planning*, Kluwer Academic, 1991.