

Sensor-based trajectory deformation: application to reactive navigation of nonholonomic robots

Florent Lamiroux Olivier Lefebvre

Abstract In this chapter, we present a sensor-based trajectory deformation process for nonholonomic robots. The method is based on infinitesimal perturbations of the input functions of the current trajectory. Input perturbation is computed in such a way that an objective function decreases and that the trajectory initial and final configurations are kept unchanged. The method is then extended to docking for wheeled mobile robots. The final configuration of the deformation process is moved to a configuration in order to make perception fit a docking pattern. The method is demonstrated on mobile robot Hilare 2 towing a trailer.

1 Introduction

Navigating multi-body nonholonomic robots in cluttered environments has been a difficult task for a long time, especially when the two following conditions are met:

1. the number of nonholonomic constraints is two or more and
2. the localization uncertainty is less than the clearance to obstacles.

Recent autonomous vehicles competing in the Darpa Urban Challenge [15] might let the reader think that the problem of autonomous navigation for nonholonomic systems is closed. However, beyond the remarkable work of integration these vehicles are the result of, it should be noticed that these vehicles would not have reached the goal without the recent advances in localization technology. Today, on-the-shelf devices compute in real time the position of vehicles with an accuracy around the meter. In this context, navigating with a margin of two meters makes the computations relative to motion planning and control much simpler. The Challenge is thus

Florent Lamiroux
CNRS/LAAS, Toulouse, France, e-mail: florent@laas.fr

Olivier Lefebvre e-mail: olivier.lefebvre.perso@gmail.com

more in the field of perception and modelling than in the field of motion planning and control.

In this chapter, we report on work aiming at addressing the navigation task for systems meeting the two above mentioned conditions. Unlike classical visual servoing methods where the state of the system is the configuration and velocity of the robot, and the input visual perception, the state of our system is a trajectory executable by the robot and the input is a flow of sensor images. This chapter is mostly a compilation of [8] and [14].

To overcome the issue of local minima arising when implementing local control laws, we initially plan an admissible trajectory from the initial configuration of the robot to the goal configuration, using classical approaches of the state of the art in motion planning [1, 11, 19, 3, 20, 12, 9, 13].

Servoing a trajectory instead of a robot state significantly reduces the issue of local minima at the cost of heavy computational load.

Following a planned trajectory can lead to collisions if

- unexpected obstacles in the environment were not in the map used for planning the motion,
- the map is inaccurate or
- the localization process is inaccurate.

To overcome these issues, [17] proposed a method that enables a robot to deform on line the path to be followed in order to get away from obstacles detected along the motion. This approach has been extended to the case of a unicycle-like mobile robot in [6] and then to the case of a holonomic mobile manipulator in [2]. In both papers, the geometry of the robot is approximated by a set of balls and no or only one very simple nonholonomic constraint is treated. None of these methods is applicable to more complex nonholonomic systems like car-like robots.

To plan and execute motions in dynamic environments, [5] developed the concept of velocity obstacles, defining the set of forbidden velocities given the velocity of the obstacles. This concept is used in [10] to perform local goal oriented obstacle avoidance. This technique is particularly efficient in environments where a lot of obstacles are moving since the velocity of the obstacles is taken into account in the avoidance strategy. However, it is based on very simple models of robot and obstacles: they all are spherical. This simplification forbids applications for multi-body mobile robots moving in very cluttered environments where the robot needs to pass very close to the obstacles.

In this chapter, we describe a generic approach of trajectory deformation applicable to any nonholonomic system. We assume that a first collision-free trajectory has been computed for the robot in the global frame. When the robot follows the trajectory, on-board sensors, for instance laser scanners, detect surrounding obstacles and map them in the global frame. If an obstacle not present in the map is detected, it can be in collision with the initial trajectory. If the localization of the robot is inaccurate, or if the map is inexact, obstacles of the map might be seen in collision with the initial trajectory by the sensors. The method we describe in this paper enables the robot to deform the initial trajectory in order to move it away from obstacles

and make the current trajectory collision-free . The current trajectory thus changes along time. As a trajectory is a mapping from an interval of real numbers into the configuration space of the robot, we naturally model a trajectory deformation process as a mapping of two real variables s and τ into the configuration space. τ can be considered as time (or more generally as an increasing function of time), while s is the abscissa along each trajectory.

The chapter is organized as follows. Section 2 defines trajectory deformation as an infinite-dimensional dynamic control system the state of which is a trajectory. In section 3, we describe an iterative algorithm controlling the deformation process to make an optimization criterion decrease. In section 4, the trajectory deformation algorithm is applied to mobile robot Hilare 2 towing a trailer. In Section 5, the method is extended to perform docking for nonholonomic robots.

2 Nonholonomic trajectory deformation as a dynamic control system

A trajectory for a robotic system is usually represented by a mapping from an interval of \mathbb{R} into the configuration space of the system. In this section, we introduce the notion of trajectory deformation as a mapping from an interval of \mathbb{R} into the set of trajectories. Equivalently a trajectory deformation is a mapping from two intervals into the configuration space as explained later in this section.

2.1 Admissible trajectories

A nonholonomic system of dimension n is characterized by a set of $k < n$ vector fields $\mathbf{X}_1(\mathbf{q}), \dots, \mathbf{X}_k(\mathbf{q})$, where $\mathbf{q} \in \mathcal{C} = \mathbb{R}^n$ is the configuration of the system. For each configuration \mathbf{q} , the set admissible velocities of the system is the set of linear combinations of the $\mathbf{X}_i(\mathbf{q})$. A trajectory $\mathbf{q}(s)$ is a smooth curve in the configuration space defined over an interval $[0, S]$. A trajectory is said to be admissible if and only if there exists a k -dimensional smooth vector valued mapping $\mathbf{u} = (u_1, \dots, u_k)$ defined over $[0, S]$ and such that:

$$\forall s \in [0, S] \quad \mathbf{q}'(s) = \sum_{i=1}^k u_i(s) \mathbf{X}_i(\mathbf{q}(s)) \quad (1)$$

where from now on, $'$ denotes the derivative with respect to s .

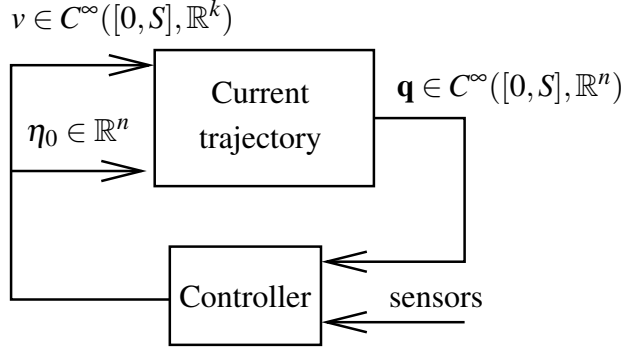


Fig. 1 A trajectory deformation process can be modelled as a dynamic control system of time τ . At each time, the state is a feasible trajectory \mathbf{q} , the input is a pair (η_0, \mathbf{v}) that uniquely defines the time derivative of the state. The trajectory deformation algorithm we describe in this chapter can be considered as a closed-loop controller that computes the input of the dynamic control system with respect to the current trajectory and a task to achieve, for instance avoiding obstacles, based on perceptual data.

2.2 Admissible trajectory deformation

We call trajectory deformation a mapping from a subset $[0, S] \times [0, \infty)$ of \mathbb{R}^2 to the configuration space of the system.

$$(s, \tau) \rightarrow \mathbf{q}(s, \tau)$$

For each value of τ , $s \rightarrow \mathbf{q}(s, \tau)$ is a trajectory. $s \rightarrow \mathbf{q}(s, 0)$ is called the *initial trajectory*. In order to keep notation light and intuitive, we use the same notation \mathbf{q} to denote configurations, trajectories and trajectory deformations. We are interested in deformations $\mathbf{q}(s, \tau)$ composed of only admissible trajectories. Such deformations satisfy the following constraint: there exists a k -dimensional vector valued smooth mapping $\mathbf{u} = (u_1, \dots, u_k)$ defined over $[0, S] \times [0, \infty)$ such that $\forall (s, \tau) \in [0, S] \times [0, \infty)$

$$\frac{\partial \mathbf{q}}{\partial s}(s, \tau) = \sum_{i=1}^k u_i(s, \tau) \mathbf{X}_i(\mathbf{q}(s, \tau)) \quad (2)$$

For each value of τ , $s \rightarrow \mathbf{u}(s, \tau)$ is the input function of trajectory $s \rightarrow \mathbf{q}(s, \tau)$. The above equation simply expresses constraint (1) for each trajectory of the deformation. As well as a trajectory is uniquely defined by the initial configuration and the input function, a trajectory deformation is uniquely defined by the initial configuration $\mathbf{q}(0, \tau)$ of each trajectory and by input functions $u_i(s, \tau)$.

By differentiating (2), we get a relation between the input variation $\frac{\partial \mathbf{u}}{\partial \tau}$ and the infinitesimal trajectory deformation when the deformation parameter τ increases:

$$\frac{\partial^2 \mathbf{q}}{\partial s \partial \tau}(s, \tau) = \sum_{i=1}^k \left(\frac{\partial u_i}{\partial \tau}(s, \tau) \mathbf{X}_i(\mathbf{q}(s, \tau)) + u_i(s, \tau) \frac{\partial \mathbf{X}_i}{\partial \mathbf{q}}(\mathbf{q}(s, \tau)) \frac{\partial \mathbf{q}}{\partial \tau}(s, \tau) \right)$$

We call respectively *input perturbations* and *direction of deformation* the following vector valued functions:

$$\begin{aligned} \mathbf{v}(s, \tau) &\triangleq \frac{\partial \mathbf{u}}{\partial \tau}(s, \tau) \\ \eta(s, \tau) &\triangleq \frac{\partial \mathbf{q}}{\partial \tau}(s, \tau) \end{aligned}$$

With this notation, the above equation becomes :

$$\eta'(s, \tau) = A(s, \tau)\eta(s, \tau) + B(s, \tau)\mathbf{v}(s, \tau) \quad (3)$$

where $A(s, \tau)$ is the following $n \times n$ matrix:

$$A(s, \tau) = \sum_{i=1}^k u_i(s, \tau) \frac{\partial X_i}{\partial \mathbf{q}}(\mathbf{q}(s, \tau))$$

and $B(s, \tau)$ is the $n \times k$ matrix the columns of which are the control vector fields:

$$B(s, \tau) = (X_1(\mathbf{q}(s, \tau)) \cdots X_k(\mathbf{q}(s, \tau)))$$

According to (3), the derivative w.r.t. τ of the trajectory of parameter τ is related to the input perturbation through a linear dynamic system. This system is in fact the linearized system of (1) about the trajectory of parameter $\tau: s \rightarrow \mathbf{q}(s, \tau)$. For a given trajectory $\mathbf{q}(s, \tau)$ of input $\mathbf{u}(s, \tau)$ and for any input perturbation $\mathbf{v}(s, \tau)$, and any initial condition $\eta_0 = \eta(0, \tau)$ we can integrate Equation (3) w.r.t. s to get the corresponding direction of deformation $\eta(s, \tau)$.

A trajectory deformation process for nonholonomic systems can thus be considered as a dynamic control system where

- τ is the time,
- $s \rightarrow \mathbf{q}(s, \tau)$ is the state and
- $(\eta_0, s \rightarrow \mathbf{v}(s, \tau))$ is the input.

2.3 Potential field and inner product

The trajectory deformation method produces at each time τ a vector η_0 and a function $s \rightarrow \mathbf{v}(s, \tau)$ over $[0, S]$ in such a way that the deformation process achieves a specified goal. This goal is expressed in terms of a scalar value to minimize over the set of feasible trajectories. The scalar value associated to a trajectory is defined by integration of a potential field U over the configuration space. We denote by $V(\tau)$ the potential value of trajectory $s \rightarrow \mathbf{q}(s, \tau)$:

$$V(\tau) \triangleq \int_0^S U(\mathbf{q}(s, \tau)) ds$$

If the goal to achieve is to avoid obstacles, as in [18, 7, 1], the configuration space potential field is defined in such a way that the value is high for configurations close to obstacles and low for configuration far from obstacles. Thus trajectories going close to obstacles have a high scalar value and trajectories staying far from obstacles have a low scalar value.

The variation of the trajectory scalar value with respect to τ is related to $\eta(s, \tau)$ by the following expression:

$$\frac{dV}{d\tau}(\tau) = \int_0^S \frac{\partial U}{\partial \mathbf{q}}(\mathbf{q}(s, \tau))^T \eta(s, \tau) ds$$

The principle of the trajectory deformation method consists in choosing $(\eta_0, \mathbf{v}(s, \tau))$ in such a way that $\frac{dV}{d\tau}(\tau)$ is negative. Let us notice that the space of vector-valued functions defined over interval $[0, S]$ is a Hilbert space, the inner product of which is defined by:

$$(f|g)_{L^2} \triangleq \int_0^S f(s)^T g(s) ds \quad (4)$$

With this definition, the variation of the trajectory scalar value along a direction of deformation can be rewritten:

$$\frac{dV}{d\tau}(\tau) = \left(\frac{\partial U}{\partial \mathbf{q}} \circ \mathbf{q} | \eta \right)$$

where \circ denotes the composition operator. Let us notice that integration is performed over variable s only. According to this expression, $\eta = -\left(\frac{\partial U}{\partial \mathbf{q}} \circ \mathbf{q}\right)$ is at equivalent L^2 -norm the direction of deformation that minimizes $\frac{dV}{d\tau}$. Unfortunately, this value of η is not an admissible direction of deformation (i.e. a solution of system (3)). A solution could be obtained by orthogonally projecting $-\left(\frac{\partial U}{\partial \mathbf{q}} \circ \mathbf{q}\right)$ over the linear-subspace of admissible directions of deformation. However, the projection of a vector over an infinite-dimensional subspace does not necessarily exist.

To overcome this problem, we will restrict the input perturbation to a finite-dimensional subspace in the following section.

3 Nonholonomic trajectory deformation algorithm

Based on the theoretical framework established in the previous section, we build in this section the trajectory deformation algorithm for nonholonomic systems. Starting from an initial admissible trajectory $\mathbf{q}(s, 0)$, the algorithm iteratively computes a sequence of admissible trajectories $s \rightarrow \mathbf{q}(s, \tau_j)$ for discretized values τ_j of τ where j is an integer. At each iteration of the algorithm, a direction of deformation $\eta(s, \tau_j)$ is generated based on the configuration space potential field U and a new trajectory

$\mathbf{q}(s, \tau_{j+1})$ is computed as follows:

$$\mathbf{q}(s, \tau_{j+1}) = \mathbf{q}(s, \tau_j) + \Delta \tau_j \boldsymbol{\eta}(s, \tau_j) \quad (5)$$

$$\tau_{j+1} = \tau_j + \Delta \tau_j \quad (6)$$

where $\Delta \tau_j$ is the discretization step. Let us notice that the above formula is a first-order approximation in τ . In the rest of this section, we describe the different steps of the algorithm. In Section 3.1, we compute $\boldsymbol{\eta}(s, \tau_j)$ by restricting input perturbation to a finite-dimensional subspace of functions. This restriction enables us in Section 3.2 to take into account boundary conditions that force the initial and final configuration of the deformation interval to remain unchanged. In Section 3.3, we explain how to compute the direction of deformation that minimizes the variation of the trajectory scalar value under constant L^2 -norm. The first order approximation 5 induces deviations of the nonholonomic constraints. Section 3.4 addresses this issue and proposes a correction of this deviation.

3.1 Finite-dimensional sub-space of input perturbations

As explained in Section 2, the control variables of a trajectory deformation process are the input perturbation \mathbf{v} and the initial condition $\boldsymbol{\eta}_0$. $s \rightarrow \mathbf{v}(s, \tau_j)$ belongs to the infinite-dimensional space of smooth vector-valued functions defined over $[0, S]$. To simplify the control of the trajectory deformation, we choose to restrict \mathbf{v} to a finite-dimensional subspace of functions. This restriction will make the boundary conditions introduced later in section 3.2 easier to deal with. Let p be a positive integer. We define $\mathbf{e}_1, \dots, \mathbf{e}_p$, a set of smooth linearly independant vector-valued functions of dimension k , defined over $[0, S]$:

$$\mathbf{e}_i : [0, S] \rightarrow \mathbb{R}^k$$

Various choices are possible for the \mathbf{e}_i 's (e.g. truncated Fourier series, polynomials,...) [16, 4, 3]. For each of these functions, we define $\mathbf{E}_i(s, \tau_j)$ as the solution of system (3) with initial condition $\boldsymbol{\eta}_0 = 0$ and with $\mathbf{e}_i(s)$ as input:

$$\mathbf{E}_i'(s, \tau_j) = A(s, \tau_j)\mathbf{E}_i(s, \tau_j) + B(s, \tau_j)\mathbf{e}_i(s) \quad (7)$$

$$\mathbf{E}_i(0, \tau_j) = 0 \quad (8)$$

where matrices A and B are defined in Section 2.2. Let us notice that unlike \mathbf{e}_i , \mathbf{E}_i depends on τ_j since system (3) depends on the current trajectory.

If we restrict $\mathbf{v}(s, \tau_j)$ in the set of functions spanned by the \mathbf{e}_i 's, that is for any vector $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_p)$:

$$\mathbf{v}(s, \tau_j) = \sum_{i=1}^p \lambda_i \mathbf{e}_i(s) \quad (9)$$

as (3) is linear, the direction of deformation η corresponding to \mathbf{v} is the same linear combination of solutions \mathbf{E}_i

$$\eta(s, \tau_j) = \sum_{i=1}^p \lambda_i \mathbf{E}_i(s, \tau_j) \quad (10)$$

Using this restriction, the input perturbation \mathbf{v} is uniquely defined by vector λ .

3.2 Boundary conditions

We wish the deformation process not to modify the initial and goal configurations of the trajectory. We thus impose the following boundary conditions:

$$\begin{aligned} \forall j > 0, \quad \mathbf{q}(0, \tau_j) &= \mathbf{q}(0, 0) \\ \mathbf{q}(S, \tau_j) &= \mathbf{q}(S, 0) \end{aligned}$$

These constraints are equivalent to:

$$\forall j > 0, \quad \eta(0, \tau_j) = 0 \quad (11)$$

$$\eta(S, \tau_j) = 0 \quad (12)$$

Equation (8) and Expression (10) ensure us that the first constraint (11) is satisfied. The second constraint (12) together with Expression (10) becomes a linear constraint over vector λ :

$$L\lambda = 0 \quad (13)$$

where L is a $n \times p$ -matrix the columns of which are the $\mathbf{E}_i(S, \tau_j)$'s:

$$L = (\mathbf{E}_1(S, \tau_j) \cdots \mathbf{E}_p(S, \tau_j))$$

Let us notice that in general, the dimension of the subspace of solutions of the above linear system is equal to $p - n$ and therefore p must be bigger than n . The problem is now to choose a vector λ satisfying the above linear constraint and generating a direction of deformation that makes the current trajectory move away from obstacles. We address this issue in the following section.

3.3 Direction of deformation that makes trajectory scalar value decrease

As explained in Section 2.3, a potential field U is defined over the configuration space. This potential field defines by integration a scalar valued function V over the space of trajectories.

Given a vector $\lambda \in \mathbb{R}^p$, the variation of the trajectory scalar value induced by direction of deformation η defined by Equation (10) is given by the following expression:

$$\frac{dV}{d\tau}(\tau_j) = \int_0^S \frac{\partial U}{\partial \mathbf{q}}(\mathbf{q}(s, \tau_j))^T \eta(s, \tau_j) ds \quad (14)$$

$$= \sum_{i=1}^p \lambda_i \int_0^S \frac{\partial U}{\partial \mathbf{q}}(\mathbf{q}(s, \tau_j))^T \mathbf{E}_i(s, \tau_j) ds \quad (15)$$

Let us define the following coefficients:

$$\mu_i \triangleq \int_0^S \frac{\partial U}{\partial \mathbf{q}}(\mathbf{q}(s, \tau_j))^T \mathbf{E}_i(s, \tau_j) ds$$

These coefficients represent the variation of the trajectory scalar value induced by each direction of deformation \mathbf{E}_i . With these coefficients, Expression (15) can be rewritten as follows:

$$\frac{dV}{d\tau}(\tau_j) = \sum_{i=1}^p \lambda_i \mu_i \quad (16)$$

Thus, if we choose

$$\lambda_i = -\mu_i \quad (17)$$

we get a trajectory deformation $\eta(\cdot, \tau_j)$ that keeps the kinematic constraints satisfied and that makes the trajectory scalar value decrease. Indeed:

$$\frac{dV}{d\tau}(\tau_j) = -\sum_{i=1}^p \mu_i^2 \leq 0$$

We denote by λ^0 this value of vector λ . Of course, nothing ensures us that λ^0 satisfies the boundary conditions (13).

3.3.1 Projection over the sub-space of boundary conditions

Equation (13) states that the set of vectors λ satisfying the boundary conditions is a linear subspace of \mathbb{R}^p . To get such a vector that we denote by $\bar{\lambda}$, we project λ^0 over this subspace:

$$\bar{\lambda} = (I_p - L^+L)\lambda^0$$

where I_p is the identity matrix of size p and L^+ is the Moore-Penrose pseudo-inverse of matrix L .

It can be easily verified that $\bar{\lambda}$ satisfies the following property:

1. $L\bar{\lambda} = 0$ and
2. $\eta = \sum_{i=1}^p \bar{\lambda}_i \mathbf{E}_i$ makes the trajectory scalar value decrease.

3.3.2 A better direction of deformation

Let us recall that Equation (5) is an approximation of order 1 with respect to τ . For this reason, $\Delta \tau_j \|\eta\|_\infty$ with $\|\eta\|_\infty \triangleq \max_{s \in [0, S]} \|\eta(s, \tau_j)\|$ needs to be small. $\Delta \tau_j$ is thus chosen in such a way that $\Delta \tau_j \|\eta\|_\infty$ is upper bounded by a positive given value η_{max} . The way the λ_i 's are chosen in (17) is not optimal in this respect. Indeed, the goal we aim at at each iteration is to make the trajectory scalar value V decrease at most for constant $\|\eta\|_\infty$. Therefore the optimal value of λ realizes the following minimum:

$$\min_{\|\eta\|_\infty=1} \frac{dV}{d\tau}(\tau_j) = \min_{\|\sum_{i=1}^p \lambda_i \mathbf{E}_i\|_\infty=1} \sum_{i=1}^p \mu_i \lambda_i$$

Unfortunately, this value of vector λ is very difficult to determine since $\|\cdot\|_\infty$ is not a Euclidean norm. Instead, we compute:

$$\min_{\|\sum_{i=1}^p \lambda_i \mathbf{E}_i\|_{L^2}=1} \sum_{i=1}^p \mu_i \lambda_i$$

This is a better approximation than (17).

The idea of the computation is to express η in an L^2 -orthonormal basis in such a way that the above sum becomes the inner product between two vectors. Let us build from $(\mathbf{E}_1, \dots, \mathbf{E}_p)$ an orthonormal basis $(\mathbf{F}_1, \dots, \mathbf{F}_p)$ using Gramm Schmidt orthonormalization procedure. Let P be the corresponding $p \times p$ matrix of change of coordinates (the j -th column of P is the vector of coordinates of \mathbf{F}_j expressed in $(\mathbf{E}_1, \dots, \mathbf{E}_p)$). If we express η in $(\mathbf{F}_1, \dots, \mathbf{F}_p)$ instead of $(\mathbf{E}_1, \dots, \mathbf{E}_p)$, Equation (10) becomes:

$$\eta(s, \tau_j) = \sum_{i=1}^p \lambda_i^\perp \mathbf{F}_i(s, \tau_j)$$

and Equation (16) becomes:

$$\frac{dV}{d\tau}(\tau_j) = \sum_{i=1}^p \lambda_i^\perp \mu_i^\perp = (\mu^\perp | \eta)_{L^2} \quad (18)$$

with

$$\mu_i^\perp \triangleq \int_0^S \frac{\partial U}{\partial \mathbf{q}}(\mathbf{q}(s, \tau_j))^T \mathbf{F}_i(s, \tau_j) ds$$

and $\mu^\perp = \sum_{i=1}^p \mu_i^\perp \mathbf{F}_i$. The second equality in (18) holds since $(\mathbf{F}_1, \dots, \mathbf{F}_p)$ is L_2 -orthonormal. At equivalent L_2 -norm, $\eta = -\mu^\perp$ (i.e. $\lambda_i = -\mu_i^\perp$) is the direction of deformation that minimizes $\frac{dV}{d\tau}(\tau_j)$. In fact we do not evaluate functions \mathbf{F}_i 's, but only matrix P . The expression of η in basis $(\mathbf{E}_1, \dots, \mathbf{E}_p)$ is given by vector

$$\lambda = P \lambda^\perp = P P^T \lambda^0 \quad (19)$$

Using expression of η in the orthonormal basis $(\mathbf{F}_1, \dots, \mathbf{F}_p)$, the expression in $(\mathbf{E}_1, \dots, \mathbf{E}_p)$ of the orthogonal projection of the above η over the sub-space of vec-

tors satisfying the boundary conditions (12) becomes

$$\bar{\lambda} = (I_p - P(LP)^+L)PP^T\lambda^0$$

Using the above optimal direction of deformation makes the trajectory deformation algorithm behave much better. It can be explained by the fact that this choice makes the trajectory scalar value decrease faster and thus is more efficient to get away from obstacles.

3.4 Nonholonomic constraint deviation

Approximation (5) induces a side effect: after a few iterations, the nonholonomic constraints are not satisfied anymore and the trajectory becomes non admissible. We call this effect the nonholonomic constraint deviation. The goal of this section is to correct this deviation. If a trajectory is not admissible, the velocity along this trajectory is not contained in the linear subspace spanned by the k control vector fields and condition (1) does not hold.

3.4.1 Extended dynamic system

To take into account this issue, for each configuration \mathbf{q} , we add $n - k$ vector fields $\mathbf{X}_{k+1}(\mathbf{q}), \dots, \mathbf{X}_n(\mathbf{q})$ to the k control vector fields of the system in such a way that $\mathbf{X}_1(\mathbf{q}), \dots, \mathbf{X}_n(\mathbf{q})$ span \mathbb{R}^n . We define the extended system as the system controlled by all these vector fields:

$$\mathbf{q}' = \sum_{i=1}^n u_i \mathbf{X}_i(\mathbf{q}) \quad (20)$$

System (20) is not subject to any kinematic constraint. A trajectory $\mathbf{q}(s)$ of system (20) is admissible for system (1) if and only if for any $j \in \{k+1, \dots, n\}$ and any $s \in [0, S]$, $u_j(s) = 0$.

In Section 2, we deformed a given trajectory, admissible for (1) by perturbing the input functions $u_1(s, \tau), \dots, u_k(s, \tau)$ of this trajectory in order to avoid obstacles. In this section, we consider an initial trajectory not necessarily admissible and we compute input perturbations that make $u_{k+1}(s, \tau), \dots, u_n(s, \tau)$ uniformly tend toward 0 as τ grows.

From now on, we denote by $\bar{\mathbf{u}}(s, \tau) = (u_1(s, \tau), \dots, u_n(s, \tau))$ the input function of system (20) and by $\bar{\mathbf{v}}(s, \tau) = (v_1(s, \tau), \dots, v_n(s, \tau))$ the perturbation of these input functions:

$$\forall i \in \{1, \dots, n\}, \quad v_i(s, \tau) = \frac{\partial u_i}{\partial \tau}(s, \tau)$$

The relation between the input perturbation $\bar{\mathbf{v}}$ and the direction of deformation η is similar as in Section 2:

$$\boldsymbol{\eta}'(s, \tau) = \bar{A}(s, \tau)\boldsymbol{\eta}(s, \tau) + \bar{B}(s, \tau)\bar{\mathbf{v}}(s, \tau) \quad (21)$$

but now, $\bar{A}(s, \tau)$ and $\bar{B}(s, \tau)$ are both $n \times n$ matrices:

$$\bar{A} = \sum_{i=1}^n u_i \frac{\partial \mathbf{X}_i}{\partial \mathbf{q}}(\mathbf{q}) \quad \text{and} \quad \bar{B} = (B B^\perp) \quad (22)$$

where $B^\perp = (\mathbf{X}_{k+1}(\mathbf{q}) \cdots \mathbf{X}_n(\mathbf{q}))$ is the matrix the column of which are the additional vector fields. With this notation, (21) can be rewritten as follows:

$$\boldsymbol{\eta}'(s, \tau) = \bar{A}(s, \tau)\boldsymbol{\eta}(s, \tau) + B(s, \tau)\mathbf{v}(s, \tau) + B^\perp(s, \tau)\mathbf{v}^\perp(s, \tau) \quad (23)$$

where $\mathbf{v}^\perp(s, \tau) = (v_{k+1}(s, \tau), \dots, v_n(s, \tau))$.

3.4.2 Correction of nonholonomic deviation

In order to make $u_{i+1}(s, \tau), \dots, u_n(s, \tau)$ tend toward 0 as τ increases, we apply the following linear control:

$$\forall i \in \{k+1, \dots, n\}, \forall s \in [0, S], \quad v_i(s, \tau) = -\alpha u_i(s, \tau)$$

where α is a positive constraint. We denote by $\boldsymbol{\eta}_1$ the corresponding direction of deformation for $\tau = \tau_j$:

$$\boldsymbol{\eta}'_1(s, \tau_j) = \bar{A}(s, \tau_j)\boldsymbol{\eta}_1(s, \tau_j) + B^\perp(s, \tau_j)\mathbf{v}^\perp(s, \tau_j) \quad (24)$$

$$\boldsymbol{\eta}_1(0, \tau_j) = 0 \quad (25)$$

3.4.3 Deformation due to obstacles

Following the procedure described in sections 3.1 and 3.3, we restrict input functions (v_1, \dots, v_k) to the finite dimensional subspace of functions spanned by $(\mathbf{e}_1, \dots, \mathbf{e}_p)$ and we compute $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_p)$ according to Equation (19). We denote by $\boldsymbol{\eta}_2$ the direction of deformation obtained with these coefficients:

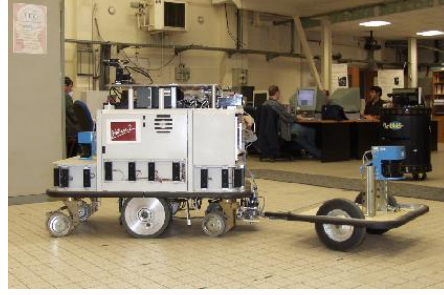
$$\boldsymbol{\eta}_2(s, \tau_j) = \sum_{i=1}^p \lambda_i \mathbf{E}_i(s, \tau_j)$$

where the \mathbf{E}_i are now solution of system:

$$\mathbf{E}'_i(s, \tau_j) = \bar{A}(s, \tau_j)\mathbf{E}_i(s, \tau_j) + B(s, \tau_j)\mathbf{e}(s) \quad (26)$$

$$\mathbf{E}_i(0, \tau_j) = 0 \quad (27)$$

Fig. 2 Mobile robot Hilare 2 towing a trailer.



3.4.4 Boundary conditions

We wish the sum of η_1 and η_2 satisfies boundary conditions (13) and (14). Again, (13) is trivially satisfied. (14) is an affine constraint over vector λ :

$$\eta_2(S, \tau_j) = L\lambda = -\eta_1(S, \tau_j) \quad (28)$$

where L is the matrix defined in Section 3.2. Following the same idea as in Section 3.2, we project vector λ over the affine sub-space satisfying (28):

$$\bar{\lambda} = -P(LP)^+ \eta_1(S, \tau_j) + (I_p - P(LP)^+L)\lambda$$

We then get a direction of deformation satisfying the boundary conditions and making the component of the velocity along additional vector fields converge toward 0:

$$\eta(s, \tau_j) = \sum_{i=1}^p \bar{\lambda}_i \mathbf{E}_i(s, \tau_j) + \eta_1(s, \tau_j)$$

Table 1 summarizes an iteration of the trajectory deformation algorithm for non-holonomic systems.

4 Application to mobile robot Hilare 2 towing a trailer

In this section, we briefly illustrate the developments of the previous section by applying them to mobile robot Hilare 2 towing a trailer (see Figure 2). We refer the reader to [8] for more details.

A configuration of this robot is represented by $\mathbf{q} = (x, y, \theta, \varphi)$ where (x, y) is the position of the center of the robot, θ is the orientation of the robot and φ is the orientation of the trailer with respect to the robot. The control vector fields are:

Algorithm : Trajectory deformation for nonholonomic systems

```

/* current trajectory = initial trajectory */
j = 0;  $\tau_j = 0$  while  $\mathbf{q}(s, \tau_j)$  in collision {

  compute  $\bar{A}(s, \tau_j)$  and  $\bar{B}(s, \tau_j)$  for  $s \in [0, S]$ 
  /* correction of nonholonomic deviation */
  for  $k$  in  $\{k+1, \dots, n\}$  {
    compute  $u_i(s, \tau_j)$ 
    compute  $v_i(s, \tau_j) = -\alpha u_i(s, \tau_j)$ 
  }
  compute  $\eta_1(s, \tau_j)$  using (24)
}

/* potential gradient in configuration space */
for  $i$  in  $\{1, \dots, p\}$  {
  compute  $\mathbf{E}_i(s, \tau_j)$  by integrating (26)
}
compute  $\frac{\partial U}{\partial \mathbf{q}}(\mathbf{q}(s, \tau_j))$  for  $s \in [0, S]$ 
for  $i$  in  $\{1, \dots, p\}$  {
  compute  $\lambda_i^0 = -\int_0^S \frac{\partial U}{\partial \mathbf{q}}(\mathbf{q}(s, \tau_j))^T \mathbf{E}_i(s, \tau_j) ds$ 
}

/* orthonormalization*/
compute matrix  $P$  using Gram-schmidt procedure

/* projection of  $\lambda$  over boundary conditions */
compute  $\tilde{\lambda} = -P(LP)^+ \eta_1(S, \tau_j) + (I_p - P(LP)^+L)\lambda$ 

/* compute and apply deformation */
compute  $\eta(s, \tau_j) = \sum_{i=1}^p \tilde{\lambda}_i \mathbf{E}_i(s, \tau_j)$  for  $s \in [0, S]$ 
 $\mathbf{q}(s, \tau_j) \leftarrow \mathbf{q}(s, \tau_j) + \Delta \tau \eta(s, \tau_j)$  for  $s \in [0, S]$ 
}

```

Table 1 Trajectory deformation algorithm: at each step, the direction of deformation $\eta(s, \tau_j)$ is computed given the current trajectory $\mathbf{q}(s, \tau_j)$ and the potential field defined by obstacles.

$$\mathbf{X}_1 = \begin{pmatrix} \cos \theta \\ \sin \theta \\ 0 \\ -\frac{1}{l_t} \sin \varphi \end{pmatrix} \quad \mathbf{X}_2 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ -1 - \frac{l_r}{l_t} \cos \varphi \end{pmatrix}$$

where l_r (resp. l_t) is the distance between the center of the robot (resp. the trailer) and the trailer connection. The inputs of the system are u_1 and u_2 the linear and angular velocities of the robot. To get a basis of \mathbb{R}^4 at each configuration \mathbf{q} , we define two additional vector fields:

$$\mathbf{X}_3 = \begin{pmatrix} -\sin \theta \\ \cos \theta \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{X}_4 = \begin{pmatrix} -\sin(\theta + \varphi) \\ \cos(\theta + \varphi) \\ -l_t - l_r \cos \varphi \\ -l_t \end{pmatrix}$$

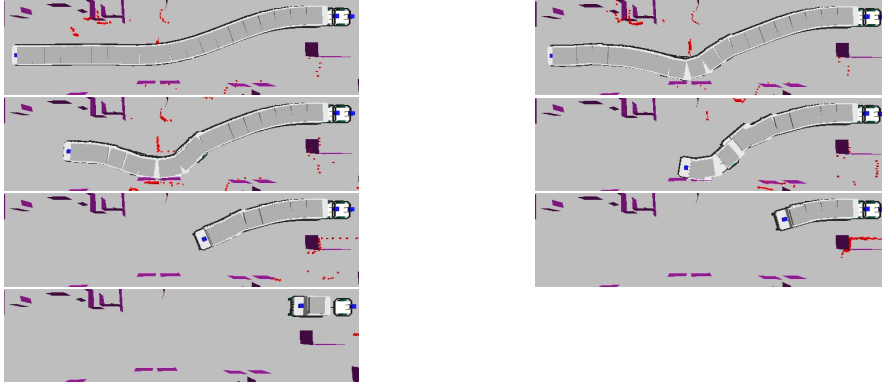


Fig. 3 A backward trajectory computed and executed by the mobile robot Hilare 2 towing a trailer. Grey dots are obstacles detected by a laser range finder mounted on the trailer. An unexpected box lies on the trajectory planned by the robot. The robot deforms the trajectory while moving and reach the goal.

The linearized system is thus defined by the following matrices:

$$\bar{A}(s) = \begin{pmatrix} 0 & 0 & -u_1 \sin \theta - u_3 \cos \theta - u_4 \cos(\theta + \varphi) & -u_4 \cos(\theta + \varphi) \\ 0 & 0 & u_1 \cos \theta - u_3 \sin \theta - u_4 \sin(\theta + \varphi) & -u_4 \sin(\theta + \varphi) \\ 0 & 0 & 0 & u_4 l_r s \varphi \\ 0 & 0 & 0 & \frac{-u_1 c \varphi + u_2 l_r s \varphi}{l_t} \end{pmatrix}$$

$$\bar{B}(s) = \begin{pmatrix} \cos \theta & 0 & -\sin \theta & -\sin(\theta + \varphi) \\ \sin \theta & 0 & \cos \theta & \cos(\theta + \varphi) \\ 0 & 1 & 0 & -l_t - l_r \cos \varphi \\ -\frac{1}{l_t} \sin \varphi & -1 - \frac{l_r}{l_t} \cos \varphi & 0 & -l_t \end{pmatrix}$$

The input perturbation is defined by truncated Fourier series over inputs u_1 and u_2 . The configuration potential field is defined by a decreasing function of the distance to obstacles in the workspace. We refer the reader to [8] for details.

4.1 Experimental results

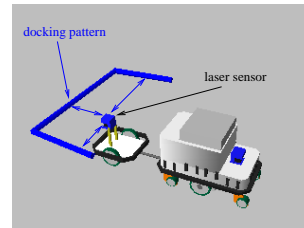
Figure 3 shows an example where mobile robot Hilare 2 avoids an unexpected obstacle detected by on-board sensors.

5 Extension to docking

The method described in the previous chapter can be extended to docking of non-holonomic mobile robots by changing the boundary condition relative to the end configuration. This is the topic of this section.

5.1 Docking task

Fig. 4 *Docking pattern.* It consists in a set of landmarks defined relatively to a sensor. In this example, the *docking pattern* is defined relatively to the laser sensor mounted on the trailer of a robot.



A docking task is a mission given to a robot that consists in following a planned trajectory and reaching a docking configuration. The docking configuration is not defined beforehand as a known robot location, rather it is specified as a set of sensor perceptions from this configuration. The set of landmarks to be perceived when the robot is at the docking configuration is called a *docking pattern*. Figure 4 presents such *docking patterns*. On each image, the docking configuration is represented relatively to the docking pattern.

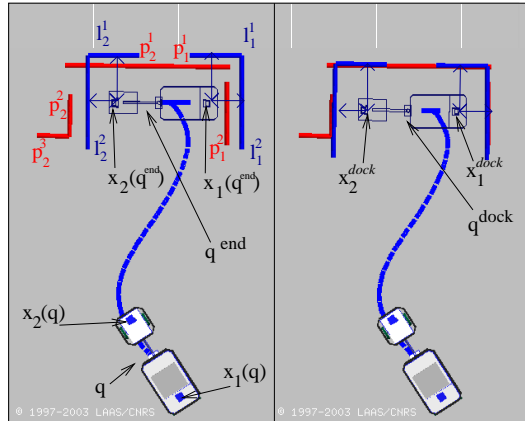
Thus a docking task takes as input:

- a collision free trajectory planned within a model of the environment
- a set of landmarks relative to the docking configuration: the *docking patterns*.

5.2 Computation of the docking configuration

In the absence of any additional information, the docking configuration is the last configuration of the planned trajectory. Otherwise, the comparison between *docking patterns* and sensor perceptions can be used to compute the docking configuration: i.e. the robot configuration where sensor perceptions best match *docking patterns*. We borrow ideas from localization and use a classical Extended Kalman filter approach to integrate this information.

Fig. 5 Example of the docking configuration. The robot scans the environment using a 2D laser scanner, from the current configuration along the current trajectory. Extracted straight line segments are matched with the docking pattern to define the docking configuration. The trajectory is progressively deformed in order to make the final configuration tend toward the docking configuration.



5.2.1 Probabilistic framework

The main steps of the computation of the docking configuration are the following.

First, the robot extract features from sensor readings and predicts from the current position of the robot how these features would be seen from the final configuration of the current trajectory. We call those the *predicted features*.

Sensor readings are modelled as Gaussian variables centered on the perfect reading for given robot and landmark positions. The predicted features are matched with the features of the docking pattern using a criterion based on the Mahalanobis distance corresponding to the Gaussian noise associated to the sensors.

The docking pattern is built from a Gaussian random configuration centered on the final configuration of the current trajectory \mathbf{q}_{rand} by evaluating the expected valued conditionally to the predicted features.

Once the docking pattern has been computed, one step of the trajectory deformation algorithm described in Section 3 is applied by changing the right hand side of boundary conditions (12) by a vector making the final configuration of the current trajectory move toward the docking configuration.

6 Experimental results

We have implemented and tested this method on a real robot. We present the results gathered after experiments in realistic scenarios.

A common scenario for a truck with a trailer is to park the trailer along an unloading platform. That is the final position of the trailer is defined relatively to the unloading platform. We have reproduced this scenario with Hilare 2 towing a trailer.

The trailer is equipped with a laser range sensor. In this experiment the landmarks are straight line segments. The *docking pattern* can be composed of any number of segments.

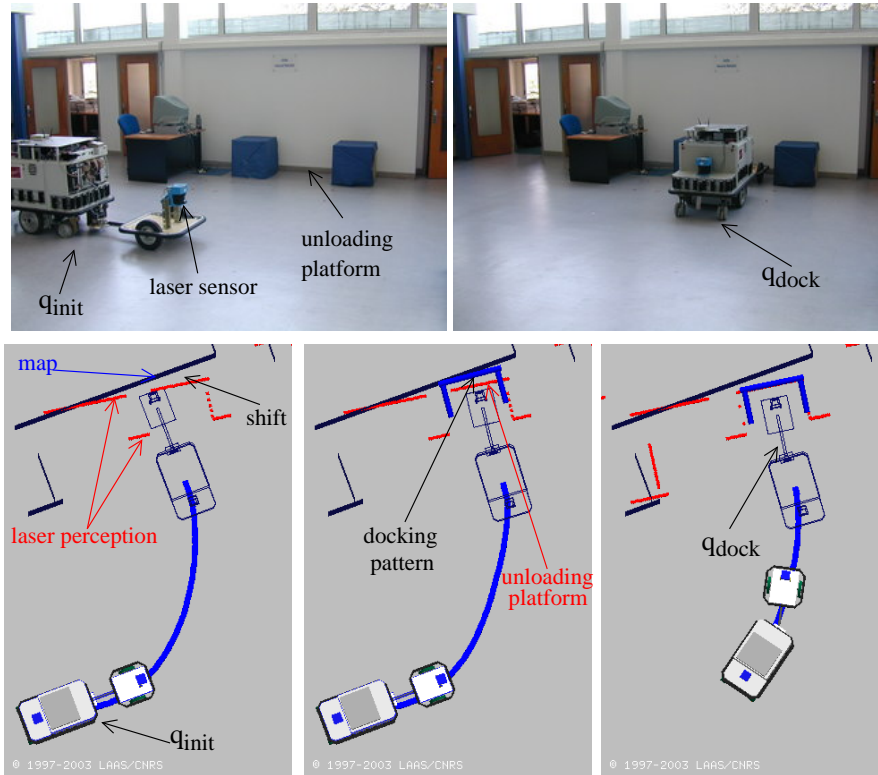


Fig. 6 A docking task: the robot is required to reach the unloading platform the shape of which is represented in bold: the *docking pattern*. The laser perception is shifted with respect to the map: it means that the robot is poorly localized. However, the robot is able to detect the *docking pattern* and to deform the reference trajectory in order to avoid obstacles and to dock at the unloading platform. The docking task is executed with respect to the perception and not with respect to the map.

6.1 Bad localization

Figure 6 represents this scenario. We see that the map does not perfectly match the perception. This is due to a bad localization of the robot. The docking configuration is anyway computed with respect to the sensor perception. The robot detects the unloading platform. Then it deforms the trajectory in order to dock at the unloading platform and to avoid obstacles. Let us notice that in this experiment the robot does not need to stop to compute the docking configuration nor to deform the trajectory. It is true as long as the docking configuration is close to the end of the trajectory.

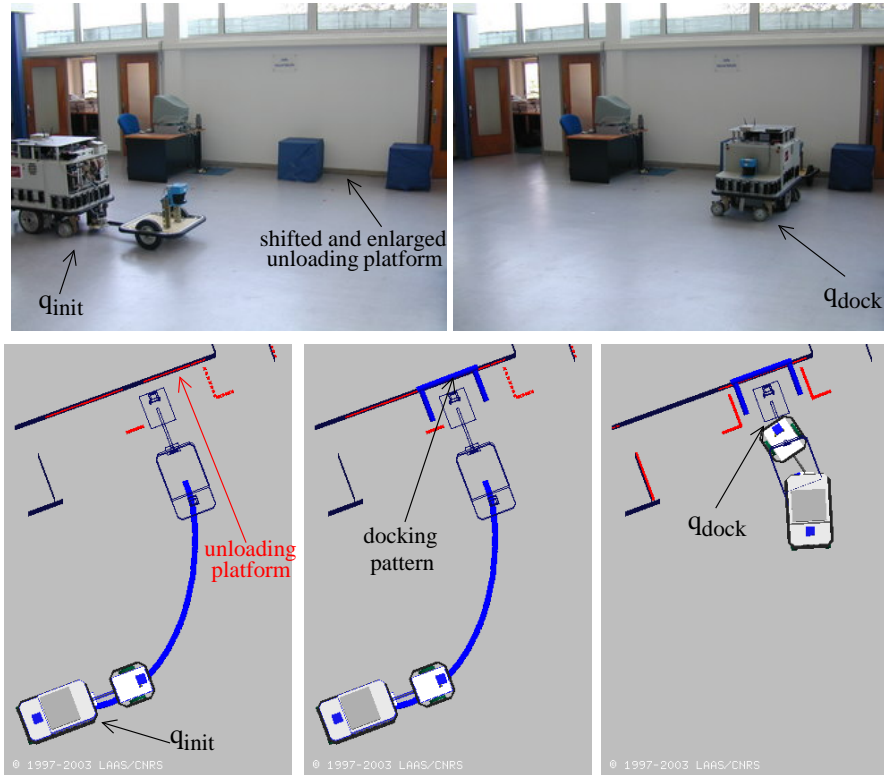


Fig. 7 The position and the shape of the unloading platform have been changed compared to figure 6. The unloading platform has been shifted to the right and it has been enlarged by 0.2 meters. The docking configuration is computed as the configuration where the *docking pattern* best fits the unloading platform.

6.2 The Unloading platform has been moved

Figure 7 illustrates the case where the unloading platform has been moved and the map has not been updated. Moreover, the shape of the unloading platform has changed: it is larger than the *docking pattern*. The matching between the perception and the *docking pattern* is robust to these perturbations and the docking configuration is still defined relatively to the unloading platform.

References

1. J. Barraquand and J.-C. Latombe. Robot motion planning: A distributed representation approach. *International Journal on Robotics Research*, 10(6):628–649, June 1991.

2. O. Brock and O. Khatib. Real-time replanning in high dimensional configuration spaces using sets of homotopic paths. In *International Conference on Robotics and Automation*, pages 550–555, San Francisco, CA, April 2000. IEEE.
3. A. Divelbiss and J. Wen. A path space approach to nonholonomic motion planning in the presence of obstacles. *IEEE Transactions on Robotics and Automation*, 13(3):443–451, June 1997.
4. C. Fernandes, L. Gurvits, and Z. Li. *Nonholonomic Motion Planning*, chapter Optimal Nonholonomic Motion Planning for a Falling Cat, pages 379–421. Kluwer Academic Press, 1993.
5. P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *International Journal on Robotics Research*, 17(7):760–772, July 1998.
6. M. Khatib, H. Jaouni, R. Chatila, and J.-P. Laumond. Dynamic path modification for car-like nonholonomic mobile robots. In *International Conference on Robotics and Automation*, pages 2920–2925, Albuquerque (USA), April 1997. IEEE.
7. O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal on Robotics Research*, 5(1):90–98, Spring 1986.
8. F. Lamiroux, D. Bonnafous, and O. Lefebvre. Reactive path deformation for nonholonomic mobile robots. *IEEE Transactions on Robotics*, 20(6):967–977, Dec 2004.
9. F. Lamiroux, S. Sekhavat, and J.-P. Laumond. Motion planning and control for hilaire pulling a trailer. *IEEE Transactions on Robotics and Automation*, 15(4):640–652, August 1999.
10. F. Large, S. Sekhavat, Z. Shiller, and C. Laugier. Toward real-time global motion planning in a dynamic environment using the nlvo concept. In *International Conference on Intelligent Robots and Systems*, pages 607–612, Lausanne (Switzerland), October 2002. IEEE/RSJ.
11. J.-P. Laumond. Controllability of a multibody mobile robot. *IEEE Transactions on Robotics and Automation*, 9(6):755–763, June 1993.
12. J.-P. Laumond, S. Sekhavat, and F. Lamiroux. *Robot Motion Planning and Control*, chapter Guidelines in Nonholonomic Motion Planning for Mobile Robots, pages 2–53. Lecture Notes in Control and Information Sciences. Springer, NY, 1998.
13. S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, May 2001.
14. O. Lefebvre and F. Lamiroux. Docking task for nonholonomic mobile robots. In *International Conference on Robotics and Automation*, pages 3736–3741, Orlando (USA), May 2006.
15. Dave Ferguson Maxim Likhachev. Planning long dynamically-feasible maneuvers for autonomous vehicles. In *Proceedings of Robotics: Science and Systems IV*, Zurich, Switzerland, June 2008.
16. R. M. Murray and S. Sastry. Steering nonholonomic systems using sinusoids. In *Conference on Decision and Control*, pages 2097–2101. IEEE, 1990.
17. S. Quinlan and O. Khatib. Elastic bands: Connecting path planning and control. In *International Conference on Robotics and Automation*, pages 802–807, Atlanta (USA), May 1993. IEEE.
18. E. Rimon and D. Koditschek. Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation*, 8:501–518, 1992.
19. P. Svestka and M. Overmars. Coordinated motion planning for multiple car-like robots using probabilistic roadmaps. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1631–1636, Japan, 1995.
20. P. Svestka and M. Overmars. Probabilistic path planning. In J.-P. Laumond, editor, *Robot Motion Planning and Control*, pages 255–304. Lecture Notes in Control and Information Sciences, Springer, NY, 1998.